DOCUMENT RESUME

ED 385 964 EC 304 157

TITLE	Project AutoMath: The Design, Development and Dissemination of an Empirically Based Drill and Practice Software Package for Facilitating Mildly Handicapped Pupil's Acquisition of Basic Math Skills. Final Report.
INSTITUTION	Instructional Programming Associates, Santa Barbara, CA.
SPONS AGENCY	Office of Special Education and Rehabilitative Services (ED), Washington, DC.
PUB DATE	30 Sep 90
NOTE	142p.
PUB TYPE	Guides - Classroom Use - Teaching Guides (For Teacher) (052) -- Reports - Descriptive (141)
EDRS PRICE	MF01/PC06 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; Computer Managed Instruction; Computer Software; *Drills (Practice); Elementary Secondary Education; Learning Disabilities; *Mathematics Instruction; *Mild Disabilities; Program Development; Program Evaluation; *Remedial Mathematics; Teaching Guides

ABSTRACT

This report describes the federally funded development and field testing of a math drill and practice computer software program for students with mild disabilities. Program features include: pre-assessment of pupil speed and accuracy, automatic placement of the pupil in appropriate problems for practice, automatic monitoring of pupil performance, and automatic adjustment of practice problems. Field testing was undertaken with junior and senior high students with learning disabilities as well as with grade 4-5 students without disabilities. A user's manual included with the report offers a quick reference to basic operations of the program, detailed information regarding each program component and associated decision-making algorithms, and a glossary of terms. A teacher's handbook provides guidelines for applying drill and practice in special and remedial math instruction, explains the theoretical and research basis for drill and practice, and includes a 40-item bibliography. A paper entitled "Microcomputer-Based Assessment of the Speed of Retrieval of Multiplication Facts, Extended Practice, and the Development of Automaticity" by Michael M. Gerber et al. is also appended. It reports on a study with elementary students who performed poorly in arithmetic. (Contains 28 references.) (SW)

The Design, Development and Dissemination of an Empirically Based Drill and Practice Software Package for Facilitating Mildly Handicapped Pupil's Acquisition of Basic Math Skills

Project AutoMath

Final Report To

U.S. Department of Education
Office of Special Education and Rehabilitation

Technology, Educational Media and Materials for the Handicapped Program
CDFA No. 18052
Improving Technology Software

September 30, 1990

Instructional Programming Associates
Sanat Barbara, California

Table of Contents

Table of Contents Cont'd

Overview

The AutoMath program was developed in response to a need for math software that provides an array of instructional decision - making functions necessary for effective use of drill and practice programs. The design of the program was predicated on the assumption that extended practice on the microcomputer will facilitate the development of automaticity in mildly handicapped learners. Previous research conducted under controlled conditions indicated that students will benefit from microcomputer practice (Hasselbring et al, 198-, Goldman & 1986, Schnorr, 1988). Yet studies conducted *in-situ* (Semmel & Schnorr, 1987), had shown that teachers do not employ math drill and practice programs in instructionally efficient ways and no reliable data supports the efficacy of math drill and practice programs in classroom settings.

Although math drill and practice programs are the most frequently used type of software in elementary and special education classes, Semmel and Schnorr found that teachers rarely employed any consistent instructional decisions in connection with software use. Adoption and use of drill and practice software tended to be uncritical, with ease of use and a minimum of disc handling the major instructional consideration. For example, few teachers actually pretested students to determine which problems were in need of practice, nor did teachers attempt to match practice problems with their own informal knowledge of the students' need for practice (Semmel & Schnorr, 1987). These considerations indicated that there was a great need for an extended practice program that could be adopted with ease for classroom and school laboratory use and which would also assume the critical instructional decision-making functions based on sound instructional principles.

This report will summarize the development, field testing and outcomes of the project to develop a math drill and practice

program for use with mildly handicapped students that address the needs described above.

Project Objectives

Specific objectives for the development of the AutoMath project were listed in the original proposal. This section of the final report will summarize the achievement of each the project's objectives

Objective 1. To develop a software program that teachers can use with minimal training or computer skills, through provision of menu-driven options and program management.

Outcomes: The final version (V 1) of the AutoMath program is managed by means of a simple menu which lists all program options. The results component of the program is also menu driven and permits the user to retrieve information about pupil performance at different levels of detail. These options range from summary performance information cumulated over all practice sessions, down to retrieval of information about the individual problem practiced during a given practice session.

Field tests in both laboratory and classroom settings with the GS version of the program indicated that teachers and students did not find any difficulty in using the program and that a single demonstration of the programs' options was sufficient for students to use the program independently.

Objective 2. To assure that the software meets the logistical and managerial demands of teachers for smooth, non-intrusive adoption into the total instructional program of the class.

Outcomes: Because of limited acquisition of Apple GS microcomputers by the schools, we were not been able to conclusively demonstrate that teachers would incorporate the program into the total instructional program of the classroom. However, we have conducted numerous field trials in both school microcomputer laboratory and in the classroom, and believe the ease of use described above, suggests that with access to the hardware, teachers will be able to adopt the program into the ongoing program of the class.

Objective 3. To develop a math drill and practice program that incorporates findings of research on the effective use of computers in the classroom.

Outcomes: The current (GS) version of AutoMath contains all of the program features originally projected for development in the proposal. This included (1) pre-assessment of pupil speed and accuracy, (2) automatic placement of the pupil in appropriate problems for practice based on the assessment outcomes, (3) automatic monitoring of pupil performance (4) automatic adjustment of practice problems based on performance data. Evaluation data has shown that the program operates as intended when used in the classroom or school microcomputer laboratory.

Objective 4. To develop a program which automatically places pupils in problems that can be most efficiently trained to automaticity

Outcomes: As intended, the AutoMath program will automatically select the appropriate problems for practice, monitor practice and delete problems from the practice set that the student does not know. Selection of appropriate problems for practice is based on an instructional algorithm which accounts for the individual pupil's accuracy and speed in response to given problems.

Objective 5. To develop a program that records and monitors pupil speed of response.

Outcomes: The AutoMath program is unique in its ability to monitor and record the users speed of response to problems displayed on the screen. The program automatically records the time interval between the display of problem on the monitor and the time the user enters the first keystroke in response to the problem. This time interval is known as the "initial keystroke latency" (IKL). Measured in milliseconds, the program records the IKL for each answer and also automatically calculates the median IKL for all problems in each lesson (20 problems). Thus, the teacher and pupil can see progress in terms of median IKL obtained for each succeeding lesson. The IKL is represented in a graph as well as in tabular summary form.

Objective 6. Develop a program that provides teachers with instructional recommendations regarding unmastered basic facts.

Outcomes: The results of assessment are available to the teacher on screen or in printout summary form and includes a list of all missed problems. The results file also provides a list of problems missed during practice sessions. Correct and incorrect problems are listed separately and the answer the pupil gave to the particular problem is also listed. This wealth of information concerning pupil error during assessment and practice simplifies error analysis. A discussion of how teachers may interpret recurrent errors in basic math facts may be found in the **Teachers Handbook** which accompanies the AutoMath program.

Objective 7. To develop a math drill and practice software package that incorporates research findings on motivational characteristics of MH pupils.

Outcomes: Field test experience with 55 students who used the AutoMath program three times a week for at least four consecutive weeks, indicated a high level of initial and sustained motivation. The programs' ability to present only those problems that the student can

answer successfully appeared to be intrinsically motivating. The students also responded well to the brightly colored graphs shown after each practice session. The graphs illustrated individual progress rates in increasing speed and decreasing error. Teachers reported no instances of pupil reluctance to attend practice session in the lab or to work on the program in the classroom.

Objective 8. To develop math drill and practice software appropriate to individual pupil needs by incorporating a pretest that automatically places a pupil in appropriate problem and type.

Outcomes: Throughout the development process its has been our goal to develop an "intelligent" program that models what a skilled classroom teacher would do if she could individually monitor and intervene with each pupil during math facts practice. Thus a series of instructional decision making algorithms are integrated into the program to contingently respond to variations in pupil behavior. At the outset of the program the student is placed in the appropriate level of difficulty for practice, based on an assessment pretest.

Objective 9. To develop math drill and practice software that monitors and reports pupil performance.

Outcomes: The program effectively monitors pupil performance in terms of speed of response and error. The problems that the pupil practices are continually revised depending on the pupil's performance. Problems that the pupil consistently gets wrong are automatically dropped from practice. Satisfactory completion of a practice set is determined through an algorithm that monitors pupil speed of response and compares speed on the current problem with speed before practice. Thus the program can determine via performance monitoring, when the pupil has attained automaticity.

Objective 10. To field test the product under normal classroom conditions and demonstrate that (1) the program is used over a sustained period of time, (2) the program results in improved pupil performance, (3) the pupil shows increased self monitoring and interest in the program.

Outcomes: Because of the limited number of Apple GS computers in individual classrooms, it was not possible to conduct an in-classroom field test. In our field test area, most teachers with microcomputers in the classroom had an Apple IIe or an Apple 2C. Thus we were not able to determine how teachers would use the software in their own classroom if it were compatible with the available hardware. However, extensive field testing was conducted in several school microcomputer laboratories that had acquired GS microcomputers. Thus we were able to demonstrate the effectiveness of the software but were not able to answer the question as to how it would be adopted in the classroom.

The evidence concerning the effectiveness of the AutoMath program in improving the performance of students using the program is strong. As a group, the thirty four elementary students who used the program showed significant gains in speed of performance after an average of 15 practice sessions (20 problems each session). The range of incorrect problems was low due to the monitoring and drop function of the program. However, the data showed significant gains in percent correct. Data on the transfer of speed and accuracy to paper and pencil timed test showed that pupils were able to transfer the improved performance to to double digit problems but did not show similar transfer when tested with double digit problems involving one or two added operations (ie, carrying or borrowing).

As may be seen in the above description of accomplishments to date, a validated, effective math drill and practice software program has been developed for use in special education and remedial math instruction. In addition to the software, A **User's Manual** has been developed as a reference tool. A **Teachers**

Handbook on the effective use of drill and practice has also been prepared to explain the theoretical and research basis for drill and practice and offers practical guidelines for the application of drill and practice in special and remedial math instruction.

To assure maximum adoption of the program and to meet the dissemination goals of the original proposal, the development of an Apple IIe version of the program was developed following the two year development phase of the project.

Program Features

AutoMath is an experimental math drill and practice program designed to aid the student in developing rapid responses to basic math facts problems. The ability to rapidly solve basic, single-digit math facts is seen by many researchers as essential to a pupil's effectiveness in solving more complex, multi-digit and math word problems.

Over the course of the development of the program, several features were added that improved and enhanced the capability of the program to make instructional decisions based on student performance. A series of internally programmed algorithms were designed to allow **AutoMath** to make "intelligent" decisions regarding a student's level placement. All of the program's algorithms were based on factors such as, but not limited to, median, mean and standard deviations of speed and accuracy. The development of these algorithms were based on contemporary research and theory concerning the development of automaticity in response to single digit math problems.

At the outset, **AutoMath** provides a pretest of the pupil's keyfinding speed to establish a target speed level. This is followed by a quick assessment of the pupil's single-digit problem accuracy which is used to determine which problems are appropriate for

practice. The pupil is automatically placed in the correct level for practice and the program continually monitors and adjusts the task in response to the pupil's performance.

A "scoreboard" and graphs showing cumulative accuracy and speed of performance are presented after each 20 problem practice lesson. The program also includes an "evaluation" component that exhaustively tests all single-digit combinations within a given operation. The program automatically places the student in the evaluation component when criterion has been achieved at level "9" problems. However, the teacher may have the student evaluated at any time. Results of pupil performance are available to the teacher at the problem, lesson, or cumulative lesson levels. The major features of the AutoMath program are summarized as follows:

Summary of AutoMath Program Features:

1. Keyboard key-finding speed test: Assesses the student's speed on typing numerals, and establishes the student's target speed of response (automaticity) level.

2. Teacher selection of the math operation that the student will practice: addition, subtraction, multiplication or division.

3. Assessment of pupil's math fact knowledge for each of the basic operations.

4. Automatic placement at the appropriate practice level based on assessment outcomes.

5. Automatic monitoring and analysis of the student's performance on practice problems.

6. Automatic advance or adjustment through all levels of an operation, dependent on the student's performance.

7. Summative evaluation of the student's mastery of an operation and/or need for further practice.

The AutoMath program provides the teacher with several types of feedback on student performance, including::

1. Student's median and fastest speed for the key-finding speed test.

2. Placement level for practice, based on assessment outcomes.

3. Number and percent correct for each lesson.

4. List of correct problems, list of incorrect problems, and median speed for each.

The **AutoMath** program also provides the **student** with several types of feedback on performance including:

1. Immediate feedback on correctness or incorrectness of responses to each problem.

2. A **Scoreboard** with summary information on the number and percent correct for each lesson, and the length of time taken to complete that lesson.

3. A cumulative graph showing percent correct for each lesson.

4. A cumulative graph showing the median (IKL) speed of response for each lesson.

AutoMath Instructional Algorithms

The algorithms used in the **AutoMath** program allow it to make "intelligent" decisions regarding a student's level of placement. All of the program's algorithms are based on factors such as, but not limited to, median, mean and standard deviations of speed and accuracy. The development of these algorithms are based on contemporary theory and research on the child's acquisition of basic math fact skills. A brief review of the algorithms used in each component of the program follows:

Assessment Algorithms

The Assessment component of AutoMath assesses a student's level for practice, based on accuracy of response, using only a small sample of single digit problems. Assessment starts by presenting problems beginning at Level 5. From there, AutoMath looks for at least a 2/3 consecutive correct ratio before testing the next level.

Once Assessment has been completed, AutoMath internally determines the student's performance at one of three options. Either:

1. The student has mastered **Level X**;
2. The student has a partial grasp of **Level X**; or
3. The student is not yet prepared for **Level X**.

AutoMath then automatically places the student at the level at which he or she has a partial grasp for practice until a mastery has been acquired.

Training Component Algorithms

AutoMath initially sets a training level determined by the student's performance on the Assessment component. Once this step has been completed, the student no longer needs to go through the Assessment component again, because the Training component automatically judges and places students in a new level (either higher or lower) depending on each student's performance. There are several criterion that AutoMath uses in evaluating a student's progress and in determining when and if a student moves up or down in level difficulty.

Step-Down Criteria

There are two ways in which a student can step or move down a Training level.

1. A student will automatically be moved down a level if there are three sessions within one level have accuracies below 80%.

2. In any given level, there are ten possible problems (in Level 1: 1+0, 1+1, ... 1+9). These problems make up the available item pool for that level. Each problem is presented twice during each session. If the same item is missed twice at anytime within a level, that problem will be dropped from the available problem pool.

   For example, a student is at Level 5. At each Training session at Level 5, the student will be presented all the items in the available problem pool twice. In the first practice session, if the student provides an incorrect answer for the problem '5+6' the first time it is presented, but provides a correct answer the second time it is presented, the item has been missed once but it still remains in the available problem pool.

   But, if on the student's next session at Level 5, the student again misses '5+6' when it is presented, '5+6' will be dropped from the available problem pool and it will not be included in the sessions that follow.

   When the number of items in the available problem pool has dropped from the original ten down to five, the program will automatically move the student down one level.

Step-Up Criteria

There are four criteria for stepping or moving up a **Training** level. All criteria must be met before a student can step-up.

1. The student must have completed a minimum of five sessions within the present level.

2. The percentage correct of the last session must be either higher than the session before, or 100%.

Both the third and fourth criterion depend on whether or not there have yet been any automatic level transitions since the student left the Assessment component and moved into the Training component.

3. The third criterion assesses the current session's lowest latency, or fastest speed. It also assesses the mean of the low latency pool. This is the pool of the lowest latencies (high latencies, or slow times, are weeded out) of all sessions at all levels.

   If there have been no level transitions, AutoMath compares the lowest latency of the current session to that of the previous session. The lowest latency of the current session must be lower than the lowest latency of the previous session for a student to advance.

   If there have been level transitions, then the lowest latency of the current session must be lower than the low latency pool mean for the student to move up one level.

4. The fourth criterion is dependent upon the adjusted session variance and the adjusted pool variance. The adjusted session variance is the standard deviation of the lowest session latency, squared. The adjusted pool variance is the standard deviation of the lowest pool latencies, squared.

   If there have been no level transitions, then the adjusted session variance of the current session must be lower than the adjusted session variance of the previous session for the program to advance one level.

If there have been level transitions, then the adjusted session variance of the current session must be lower than the adjusted pool variance for the program to step-up to the next level.

Evaluation Algorithms

Once a student has completed the Evaluation, AutoMath automatically "evaluates" that student's progress. A completed Evaluation ends by either graduating a student because he or she has moved through and mastered each level in a given operation and is ready to move on to another operation; or, if the student is not ready for graduating, the Evaluation component ends by reassessing the student's appropriate level and automatically reassigning to that level.

The Evaluation component of AutoMath is centrally concerned with the time criterion source (the time limit in which a student should try to answer the questions presented). Note that the time criterion source is taken from the student's performance in the Training component. The source is equal to the established mean speed plus 2 times the established standard deviation. If the time criterion source derives from the student's performance in the Keyboard Test, the source is equal to the established median speed of latency plus 2 times the established standard deviation.

Graduating

There are four criteria that the student must meet before AutoMath considers the student to have successfully completed all practice trials in a given operation, ie, "**graduated**".

1. All the Evaluation questions must be answered. This means that the student must repeat the Evaluation component until AutoMath tells the student that his or her evaluation has been completed. This could mean that the student would have to

repeat the Evaluation component up to ten times, depending on performance, before the Evaluation has been completed.

2. The percentage correct of all Evaluation questions must be at least 80%.

3. The lowest latency of the Evaluation component must be less than the time criterion source.

4. The adjusted evaluation variance must be faster than the time criterion source.

Assignment to a New Level

If upon completing the Evaluation component, the student has not fulfilled each of the above criterion, the student's appropriate level is automatically reassessed and the student is assigned to a new level. To understand how this is accomplished, first note that each problem that is presented in the Evaluation component is necessarily answered in one of three ways:

1. The student answered the problem correctly and within the time limit set by the time criterion source. The answer is therefore termed "quick and correct."

2. The student answered the problem correctly, but not within the time limit. The answer is therefore termed "slow, but correct."

3. The student answered the question incorrectly. Time is therefore irrelevant.

To reassign the student to a new level based on performance in the Evaluation component, AutoMath looks at the Evaluation problems that were answered "slow, but correct." AutoMath then takes the latencies of these questions and the latencies of their reciprocals and forms a pool of latencies. In other

words, if '5+9' was answered "slow, but correct," AutoMath places the latency of '5+9' in the latency pool, as well as the latency of the question '9+5'.

These latencies are ranked from fastest to slowest. That ranking is factored with the number of problems at each level that were answered "slow, but correct" and uses that factoring to determine the student's appropriate level.

Formative Evaluation

Formative evaluation of the AutoMath software was conducted in two phases. In the Summer and Fall of 1988, we conducted extensive informal trials with selected learning handicapped students to observe program operation and student response to various aspects or components of the program. Following program revisions, we initiated Phase 2, consisting of extensive field-based trials with three subsamples, in the Spring of 1989. All learning handicapped and remedial math students who participated in both series of trials attended schools in Santa Barbara county. Self contained classrooms for learning handicapped pupils at a junior high and senior high school were selected to form the first and second subsamples of the field trials. A local elementary school with both remedial math and learning handicapped upper elementary students was selected as source for the third subsample. Thus, subsamples of students were selected to establish both the effectiveness and validity of the AutoMath software program.

It was anticipated that program effectiveness would be demonstrated by increases in the pupil's speed of response to single digit problems and by increased accuracy across practice sessions. Program validity would be demonstrated by improved speed and accuracy on a paper and pencil transfer test. This measure should indicate whether improvement in performance due to practice on

the AutoMath software transferred to conventional classroom media.

Phase 1: Informal Trials

Subjects. Eight students identified by local schools as having severe learning handicaps (e.g., learning disabilities) participated in the first phase of our formative evaluations of the AutoMath program.

Setting and general procedures. All trials were conducted by project staff with one student at a time in an informal setting. During this phase, project staff were interested specifically in evaluating the program-user interface of a prototype, including students' ability to understand screen-presented information (readability) and graphics (interpretability), students' reactions to appearance of screens and tempo of screen changes or item presentation, keyboarding skills implicitly demanded by program, and so on. Each trial was open ended, permitting systematic and deliberate observation and interview.

Analysis and results. Each student trial was video taped. Oral reports on each trial were then presented to the project staff along with video tapes for general discussion by project staff. Based on these data and discussion, we made several specific modifications and improvements in the user-program interface.

Independent of these programming and user-program interface issues, we also conducted a preliminary analysis, on a case by case basis, of the effects of our practice algorithms on gains in speed of retrieval. That is, we wished to assure ourselves that the general approach we were making to speeding retrieval of basic math facts through repeated practice was sound. Indeed, across all cases we were able to demonstrate an increase in speed (i.e., decrease in response latency) on practiced items. While individual records of performance were not strictly comparable due to the

nature of our informal, individualized examination of students' use of the software, two generalizations were suggested by these trials. First, speed gains occurred reliably when students practiced known items. This was, of course, consistent with our expectations based upon previous work, but it made clear to us that the program needed to automatically adjust itself to incorrect responding that had not been anticipated by either the teacher or program-determined initial placement. The program was not designed to teach towards accuracy, only towards speed. Thus, when items were practiced that were not yet reliably correct, speed of response was both more highly variable and much slower than for practice items that were reliably correct (i.e., "known"). Thus, we built algorithms (noted below) that monitored error rate as well as speed, and adjusted practice problem sets and speed standards accordingly.

Second, we observed a trend across cases for a 500 millisecond difference between the fastest and the median response latencies when students were asked to simply find and press keys corresponding to randomly presented one-digit numbers on the screen. These "initial keypress latencies," or IKLs, represent no, or very small, cognitive load compared to responses that represent some arithmetic process. That is, we interpret these latencies in terms of perceptual-motor speed factors that vary by individual. We were intrigued with this result because we had decided to include in the AutoMath program's algorithms measurement and use of IKLs as reasonable proxies for obtainable levels of "automaticity." Although we believed that recognition of a digit, keyboard scanning (allowing for familiarity differences), and key press represent process components that should already be rather automatic, we were surprised both at the degree of individual variation, and yet the degree of similarity in fastest IKL - median IKL differences. Apparently, *even the simple processing demands implied by IKLs do not necessarily produce consistently optimal speed of response.* As a consequence, and as noted below, we adjusted AutoMath's internal algorithms so that IKL measurement was not only improved, but also relegated only to a *first approximation* to

expected speed of response. Instead, we determined that continued sampling of practiced responses by the program would produce an increasingly precise estimate of individual speed potential.

Phase 2: Field Trials

Subjects. Subsample 1 and 2, respectively, consisted of junior and senior high school learning handicapped students attending self-contained day classes. Subsample 3 consisted of fourth through sixth grade students from the same school. Each sample was drawn from three different schools in three moderately sized, mid-income suburban California school districts. Approximately six students participated as subsample 1 and an equal number as subsample 2. Thirty-four learning handicapped and non-handicapped students who were relatively low performing in math compared to their classroom peers and who were perceived as needing "extended practice of basic mathematics facts on a computer to increase their speed" were nominated by their teachers to participate as subsample 3.

About half of subsamples 1 and 2 were boys. The total number of students in subsample 3 was also divided about evenly between boys (18) and girls (16). Also in this subsample, a small number of students were non-native speakers of English (3); seven (7) were receiving special education resource program services . Standardized mathematics test scores available for inspection validated teacher's judgements (Mean = 45%ile , SD = 24) of need for practice in this subsample.

Setting and general procedures. The settings and procedures of evaluation for all subsamples were equivalent although, because they tended to be lower in achievement, we monitored more closely and modified conditions of practice more readily with those students with learning handicaps in subsample 2.

At the outset of the project, teachers were given the option of selecting the operation they wanted each student to work on. Subsample 1 worked on multiplication, as did most of subsample 2 (two students worked on division). With the exception of one fourth grade boy whose teacher placed him in subtraction, all other students in subsample 3 were placed in multiplication by their teachers.

All practice trials were held in a normal size classroom. Subsample 3 practiced in a classroom used by the participating school as a microcomputer laboratory. The laboratory consisted of sixteen Apple IIGS microcomputers with standard features arranged as a series of open work stations. Students in all subsamples were scheduled to participate in the automaticity practice trials three times each week with each trial consisting of two lessons of twenty problems. Trials typically lasted about ten minutes. All of the students had had a variety of experiences using computers in this laboratory as part of their normal school program and, therefore, had considerable experience in using commercial software. Each student used his/her own copy of the practice program provided on 3.5" floppy diskettes.

Non-Computer math performance. Several measures of student performance on basic (single digit) math problems were obtained using conventional paper and pencil tests. Prior to beginning computer practice trials, all students were given an untimed test of single digit number facts. All one hundred combinations of 0 through 9 were included in the 100 problem test. The purpose of this test was to determine the students knowledge of the number combinations (multiplication in this instance) and the appropriate level for practice. It was expected that this power test would yield the same information as the assessment component of the AutoMath program and thereby establish the validity of the assessment- placement paradigm of the software.

A second pretest was administered to assess the speed of response to problems of increasing procedural difficulty. The speed probes consisted of three sections of 30 problems each. The first section included single digit problems with no additional procedures. The second 30 problem set had a mix of single and double digit problems requiring one additional procedure (ie, carrying) for a correct solution. The third 30 problem set had double digit problems with two additional procedures needed for the correct answer.

The untimed power test was administered as a pretest before the computer practice trials were begun. The timed probes were administered as a pretest and then repeated after every six practice trials. Thus in addition to the pretest, there were timed probes administered after the 6th, 12th, 18th and 24th training sessions.

The speed probes were designed to determine whether the pupil's increased speed on single digit problems on the computer would effect an increase in the speed of response to double digit problems and problems with added operations in a paper and pencil transfer test.

Training trials. Each student was scheduled at least three times per week for a practice session using the AutoMath software. Each practice session consisted of two lessons of twenty problems each. Thus, we targeted completion by each subject of six lessons per week over a four week period. for a total of twenty four practice trials.

At the outset of the practice sessions, each student completed a key-finding test that was used to establish a target response latency for practice. The median key-finding speed obtained by each subject on the key-finding task, was used as criteria for establishing that the subject had approached or developed automaticity. The key-finding task consisted of the display of forty randomly generated digits (0 through 80). The key-finding task

was followed by the AutoMath Assessment component which was designed to set the appropriate level for practice. The program automatically selected the practice level for each student, based on the results of the assessment.

Subsamples 1 and 2 shared one or two computers available to their teachers. The laboratory in which the study of subsample 3 was conducted contained 16 Apple GS microcomputers and was located in the school from which the subjects were drawn. The lab was used by all classes in the school on a scheduled basis. Typically, two of the three weekly sessions were conducted during computer time for the subjects own class. Students not participating in the study used the lab for a variety of other programs, working alone or in dyads. As a result, the lab, although spacious, was occasionally noisy. Transfer tests and speed probes were administered by a project assistant at a table in the laboratory.

Program features. The program used in this study was specifically designed by the authors to enhance development of automaticity of basic math skills. Generally, the visible aspects of the program consist of vertically oriented multiplication facts displayed on the screen in large (i.e., about 48 pt.) font. Students press the number key, either on the top keyboard row or on the number pad, that corresponds to the product. Screen presentation is timed for decisions internal to the program, but exposure time is not arbitrarily limited. Students press the RETURN key when they are ready for the next multiplication.

The program itself consists of three major related components: a placement assessment, a series of practice item sets, and a series of measurement and decision algorithms.

Placement assessment. There are two major elements to the computer-based placement assessment. The first element samples basic facts from a 10 x 10 matrix of addition, subtraction, multiplication, or division facts. In the present study, the sampling matrix consisted of all multiplication facts with multipliers and

multiplicands between 0 and 9. The sampling algorithms employed by the program "test" logical likelihood of students' accuracy on facts by sampling items with the same and incrementally larger or smaller multipliers. The output of this sampling process is a "placement" level, or initial practice set.

The second element of the placement pretest has students respond to display of single digit numbers by pressing the key for that number. This produces the "initial keypress latency" (IKL), a measure intended to reflect a response requiring little information processing. The IKL established a recognition, scanning, and motor speed baseline used to "calibrate" other latency measurement algorithms.

Practice item sets. Following completion of the assessment, the program scheduled practice for students at one of ten "levels." Levels were defined as all items from the multiplication matrix that had the same multiplier, and sequenced ordinally. Thus, items like 5 x 3, 5 x 7, 5 x 9, etc. are all pooled at the same level, while items like 4 x 3, 4 x 7, and 4 x 9 are pooled at a different, "lower" level. Each practice set consisted of two, random presentations of each item minus any item that students repeatedly missed on the initial assessment.

Measurement and decision algorithms. Keypress latencies, in milliseconds, were measured by the Apple IIGS internal clock as the time from initial screen display to final keyed response. Latencies for each response and a calculated median and fastest latency of response during each practice trial were stored both as data and as variables for subsequent program decision-making.

The core algorithms operating in the program concern decisions about whether demonstrated speeds of response are likely to be as fast as possible -- that is, automatic. When an affirmative decision is made, students progress to the next higher

level of multiplication items. When the decision is negative, another practice trial at the same level is internally scheduled. This decision process occurs within the program after every completed trial. Thus, the actual number of practice trials required by any given student at a given level varies as a function of that student's unique history of performance as measured by the program.

In essence, the automaticity decision algorithms represent tests of four criteria. First, the program questions time (number of trials) in level. Retrieval time as well as computer measures of time are subject to error due to transient influences on attention. Therefore, students must complete at least five trials.

Second, the program checks to make sure that 80% or better accuracy has been maintained. Students must practice correct responses and only latencies for correct responses can be meaningfully interpreted in terms of developing automaticity.

Third, the median latency for correct responses from the current practice trial must be as fast or faster than the median of pooled latencies composed from the lower 50%ile of previous practice trials. Because attention, scanning, and motor fluctuations influence the right (i.e., high) hand portion of latency distributions, a comparison standard is created by incrementing only from latencies on the left (i.e., low) side of the distribution.

Fourth, the standard deviation of latencies in a given trial must be as small or smaller than the standard deviation of pooled latencies composed from the lower 50%ile of previous practice trials. If individuals differ meaningfully, automaticity represents a limit or boundary condition that is difficult to establish on the basis of central tendency measures. Reasonably, such a limit would be represented not only by unchanging mean or median latencies, but by high and unchanging similarity (i.e., leptokurtosis) among latencies.

Thus, in this study students were considered "automatic" when they simultaneously could demonstrate sufficient practice, sufficient accuracy, and speed gains associated with reduction of variability.

Research Design and Measures

Individual differences in practice level, number of practice trials required, and practice curves for latencies were all expected and theoretically pertinent in the present study. Therefore, each student served as their own control; initial and early latency measures served as baselines against which speed displayed in later trials were compared. To provide a degree of comparability, however, all students were scheduled to practice a total of 24 lessons even if they practiced at different levels for different lengths of time.

The program measured, stored, and statistically analyzed keystroke latencies and accuracy for each item in each trial for later retrieval. Keystroke latencies were measured as time elapsed between screen display and final keyed response rounded to one-hundredth of a second.

In addition, a series of three, thirty problem, paper and pencil probes for transfer were administered as a pretest and following the sixth, twelfth, eighteenth, and twenty-fourth practice trials. Probes consisted of two x one digit multiplication problems that varied on the number of procedural steps (i.e., carrying, regrouping) necessary to compute products. The first probe set required no procedural steps, the second required one step, and the third required two procedural steps.

To assure that students were, in fact, reliably accurate in multiplication, and to test the validity of the placement assessment component, all students were administered an untimed paper and

pencil test of all one hundred combinations of single digit number facts prior to beginning computer practice trials.

Results

Effectiveness of the AutoMath software was tested by posing the following empirical questions:

1. How many of the students responded more rapidly to practice problems and how fast did they get?

2. Did students' accuracy improve over practice trials?

3. How did student performance on practice trials relate to performance on paper and pencil transfer tests?

Pretests. Pretesting showed that students participating in this study were reliably accurate (Mean = 94%, SD = 9%) on the hundred basic multiplication facts and, therefore, suitable candidates for automaticity training. Initial keystroke latencies (keyfinding without computation) averaged 1.50 seconds (SD = .26). Practice results for the 34 students varied as expected. However, overall results showed a generalized pattern of low error and decreased keyboard response latency over trials. No performance differences were attributable to gender or students' special education status. This section will present a summary of these general findings.

Accuracy. Most students were initially placed at the "nines" level of practice. A very few were reassigned to a lower level when accuracy seemed low or too variable. All students tended to display high rates (i.e., over 80%) accuracy as intended by the structure of the program. Early trials showed greater variability than later trials, but variability decreased and accuracy increased over trials.

Speed Gains. Practice curves for all students showed decrease in latency over trials, although level, rate, and pattern of decrease varied across students. Approximately one-third of the students failed to meet the criteria for automaticity after 24 trials, although these students all showed gains in response speed.

A one factor, repeated measures analysis of variance ($F = 17.61$, $df = 23$, $p = .0001$) showed statistically significant gain in median response speed over trials. Scheffe's procedure for post-hoc comparisons showed significant differences between the first trial. the fourth trial, the sixth trial, and all subsequent trials after the eighth practice trial. Median latency (averaged across students) on Trial 1 was 2.33 seconds (SD= .52). Median latencies decreased to 1.67(.50), 1.46 (.41), 1.32 (.40). and 1.26 (.35) on Trials 6, 12, 18, and 24, respectively. Thus, by trial 12, median fact retrieval speed was essentially the same as prepractice keyfinding (IKL) speed without computation. Figure 1 shows the decline in median latency over all 24 trials. While students as a group showed a median decrease in latency of about one second by Trial 18 (.70 seconds by Trial 6), individual gains by Trial 24 varied from half a second to almost two seconds. Figure 2 is a histogram showing the distribution of students who displayed this range of median speed gains.

Transfer. A one factor, repeated measures analysis of variance showed that students produced statistically significantly more correct answers per minute ($F = 26.06$, $df = 4$, $p = .0001$) in later compared to pretest probes of simple multiplication with no procedural steps. Scheffe's procedure for post-hoc comparisons showed that speed on the third and fourth probes (i.e. after trials 18 and 24) were greater than pretest.

A similar analysis was performed for each of the other probe tests. For one procedural step probes, solution speed did not significantly increase as a function of probe occasions ($p=.08$). However, a significant change in speed did occur on the two

procedural step probe items ($F = 3.78$, $df = 4$, $p = .007$). Scheffe's procedure for post-hoc comparison showed that speed of producing correct solutions was significantly faster on the last (i.e. after trial 24) compared with the first (i.e., after trial 6) probe. Figure 3. presents a graphical summary of pretest and probe outcomes.

Case studies. Random cases from all samples were selected for closer quantitative and qualitative analysis. Figures 4, 5 and 6 depict three illustrative cases, one from each subsample. These students were practicing multiplication, addition, and division, respectively. The top graph in each figure shows changes in mean and standard deviation of response latencies for correct problems over trials.

Because interpretation of changes in speed (i.e., latency) depend upon how one defines automaticity, we also sought to model the relationship between size of numbers in a calculation and latency. In theory, automaticity represents a speed limit that is approached when answers are directly retrieved with little or no attention required. That is, responses are *theoretically* automatic when no specific element of a given arithmetic problem -- e.g., number size -- demands additional processing attention, and therefore time. Said differently, students who do not directly retrieve, are not yet automatic, require some amount of time to calculate a correct answer. There is much evidence that in mental addition, at least, calculation occurs by means of a counting strategy. Each "count" requires both attention and time. Therefore, while students are not automatic in addition, their observed latencies of response will be a monotonically increasing function of the "size" of the numbers being added (i.e., either addends or sum).

The bottom graph shows changes in intercept and slope from linear regression estimates calculated for each trial. The case depicted in Figure 5's bottom graph illustrates this approach, because the student was practicing addition, the best modeled of the four arithmetic operations. In this case, latency is regressed on

the smallest addend, thus modeling a "min" strategy. In a min strategy, students recognize and encode the largest addend (in this case, nine) as a cardinal number and count forward as many counts as are indicated by the smallest addend. Thus, they have minimized the counting effort -- hence the term "min strategy -- and their latency of response will be a function of how many counts were needed.

Similar models are presented in Figures 4 and 6 for multiplication and division, respectively. Extant research offers less clarity about what aspects of these operations are attention demanding before students become "automatic." In the Figures, we have modeled sensitivity to multiplier and quotient size, respectively. Sensitivity to the multiplier (i.e., the bottom number in columnar multiplication) can be construed as analogous to use of a counting strategy in addition. In this case, students "count" in multiples to obtain correct answers, as in 5...10...15...20 to obtain the product of five times four. On the other hand, we have modeled quotient sensitivity because non-automatic division -- e.g., 20 divided by five equals ? -- requires that students convert to a multiplication problem of the form: ? times 5 equals 20. Approached in this way, students must begin counting by multiples of five until they reach 20, keep track of the number of multiple counts produced, and convert that cardinal number into the ordinal number representing the quotient.

The bottom graph in each Figure represents the results of this modeling process. Because latencies are expected to approach perceptual-motor speed limits when the number mentally manipulated is zero, intercepts may be interpreted as estimates of this speed Thus, *theoretical* automaticity occurs when estimated *intercepts* are as low as possible and *slopes* are very close to zero. For example, in Figure 4, declining and stabilizing standard deviations indicate that the decrease in mean response latencies over trials represents a real rather than artifactual trend. Interestingly, in this case as in others we examined, slopes relating

problem size to latencies tended already to be quite low during initial trials, and generally maintained or decreased slightly from those levels. Intercepts, however, tended to show a marked decline over trials, indicating that gains in response speed were more related to gains in perceptual-motor efficiency than to gains in computation or retrieval. Figure 6 (a high school student in subsample 2) portrays a similar profile for division practice.

Figure 5, depicting a junior high school student practicing addition in subsample 1, shows a similar pattern of performance despite a few vivid examples of distraction and inattention around the seventh and eleventh trials. Students in this subsample were more severely handicapped than those in other samples and were characterized by their teacher as highly distractibility. Thus, our data showed that not only was AutoMath an effective means of promoting practice and speed in simple arithmetic for these students, but also it provided a very sensitive measure of distraction episodes. We were pleased, therefore, that the design of the program insulated important decisions about level and length of practice against this type of statistical "noise".

Discussion

In the evaluation study we specified an operational definition of automaticity of simple math fact retrieval in the context of a computer program and attempted to estimate its effectiveness for intermediate elementary students who performed poorly in arithmetic in two ways. First, we observed effects of practice on latency of correct multiplication responses in thirty-four individual cases. Second, we attempted to assess what facilitative effect might exist on multiplication problems that required increasing procedural, as opposed to computational, effort as a function of speeded retrieval of simple facts.

Despite the fact that individual patterns of gain differed, especially with respect to inter-trial variability, almost all students showed gains in median latency of 500 milliseconds or more. Median correct response latencies in later trials compared favorably with median latencies for simple key finding (IKL). Thus, students were retrieving and finding appropriate keys to respond in about the same time it took them earlier to simply find keys corresponding to numbers displayed randomly on the screen.

One unexpected finding during case analysis, however, was that a major portion of speed gain in many cases was attributable to a perceptual-motor component of response rather than computational component. This effect is apparently not attributable to unfamiliarity with computers or keyboard modes of responding. In fact, most of our field trials subjects had considerable experience using computers and keyboard responding. Moreover, the fact that speed gains during practice were associated with transfer in accurate speeded performance to paper and pencil versions of the same problems as well as to multidigit problems demonstrates that AutoMath did promote faster, accurate responding.

Our data can be interpreted to suggest, though, that theoretical specifications of automaticity may be distinguished from the type of effectively speeded performance produced by AutoMath. Evidently, while teachers were correct in nominating these students as *slower* than peers in simple arithmetic, they could not differentiate those whose slowness was based on inefficient computation strategies from those who were slow in perceptual-motor components of the task. Component processes in computation execute at speeds measured in hundreds of milliseconds. Thus, teachers, or researchers for that matter, have not been able previously to separate perceptual-motor, computation, and retrieval speeds as we have done in this field study.

It is clear from our field trials with subsample 3, for example, that speed gains on simple facts correlated with gains in speed of accurate solution of more complicated multiplication when that multiplication required no procedural knowledge or effort. The evidence from both one step and two step procedural probes might be interpreted to show a facilitative effect of speeded retrieval of basic facts, but such a conclusion is premature. All probes were repeated a total of five times. Computational aspects of these tasks may have improved as a simple function of repeated exposure and practice rather than as a consequence of developing automaticity of fact retrieval. If this explanation is valid, it would be expected that "no step" problems would improve at a faster rate than "step" problems. The data were consistent with this expectation. For although the rate of correct problem solving was found to be higher between the 6th and 24th trials on the two step transfer problems, the practical significance of this outcome is diminished by the fact that students were working at an extremely slow rate. In general, when employment of added non-computational procedures were necessary, gains in speed of correct responding were small.

Dissemination

Several measures were undertaken to disseminate information about the AutoMath program to the relevant professional community. Over the course of development, both preliminary and summative field test results were presented at professional and scholarly meetings; CEC Conference on Special Education Technology, December 1988, Reno, NV, and the American Educational Research Association, April 1990, Boston MA. A paper reporting the role of microcomputer assessment and training in the development of automaticity has been submitted to the *Journal of Special Education Technology.*

As part of our dissemination strategy, a package was designed and produced. It included all changes in the program indicated by the field test outcomes. The AutoMath program was packaged into a three ring binder containing three components: (1) AutoMath Users Manual, (2) AutoMath Teacher's Handbook and (3) AutoMath Master and Preview Disks. Both an Apple GS and Apple 2C/2E versions of the program were produced.

It is our intent to use these prototype packages to actively pursue appropriate dissemination/ publication outlets. Our field test results and our experience in working with the program suggest that AutoMath is unique in the overcrowded but undistinguished field of math drill and practice programs. The program is highly utilitarian and "teacher friendly" in its potential application in the classroom. At the same time, its capability for precise and continuous measurement of student performance make it highly useful for the most sophisticated of teachers and math coordinators. The program also has important features that make it an excellent research tool for the study of children's' development of basic math fact proficiency.

FIGURES

Figure 4. Mean latencies, standard deviations, and regression coefficients for an elementary LH student.

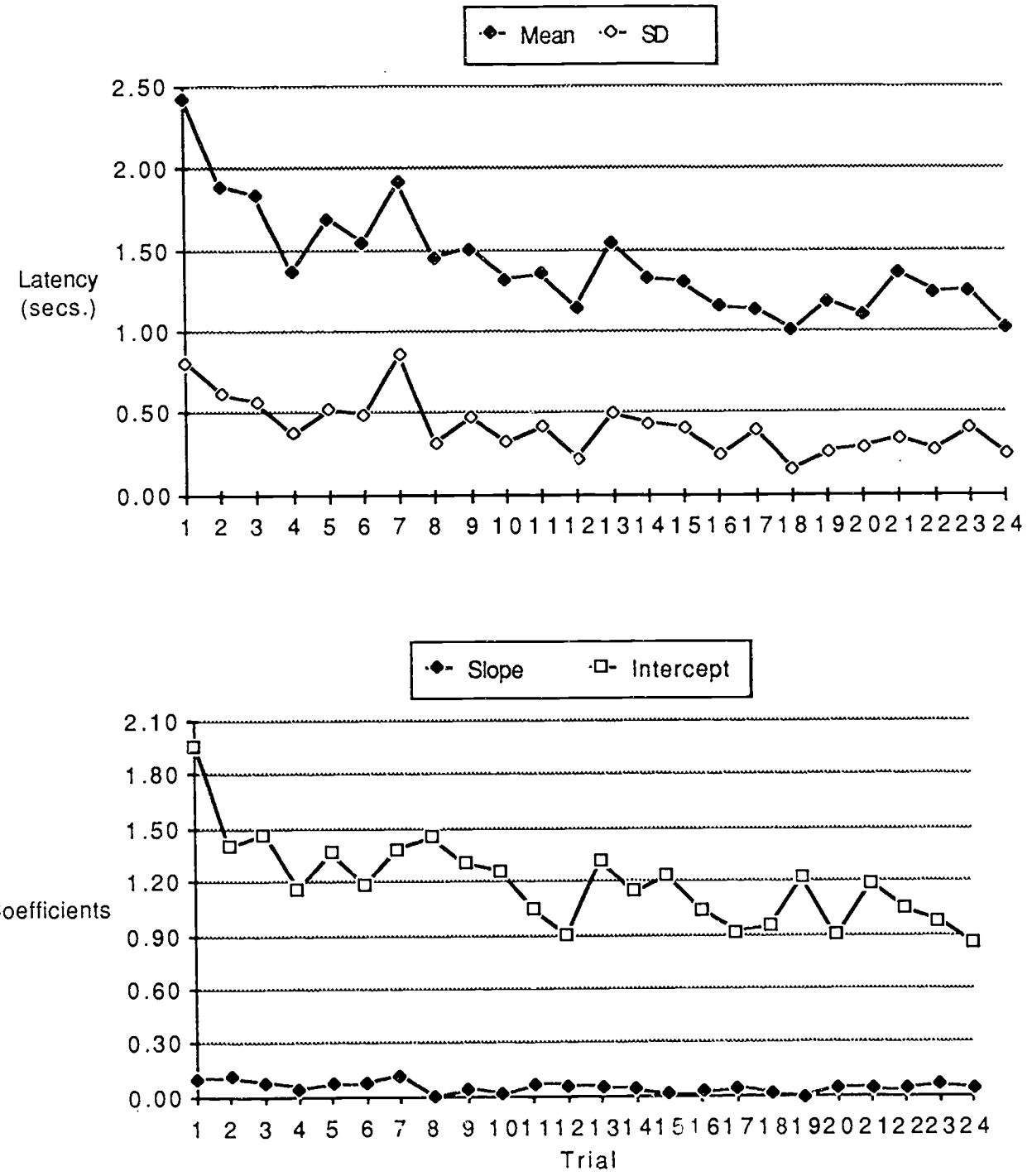


Figure 5. Mean latencies, standard deviations, and regression coefficients for a junior high school LH student.

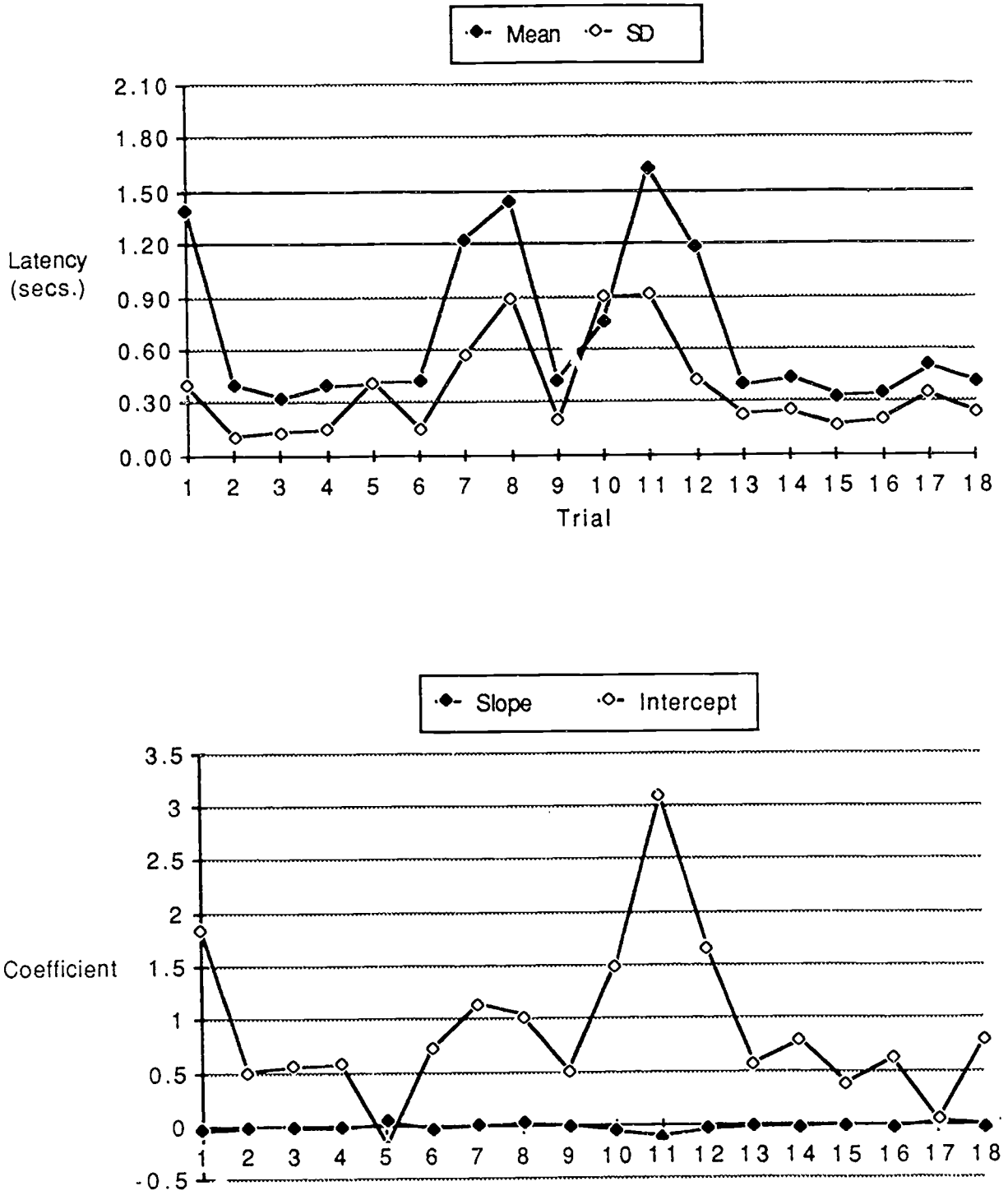
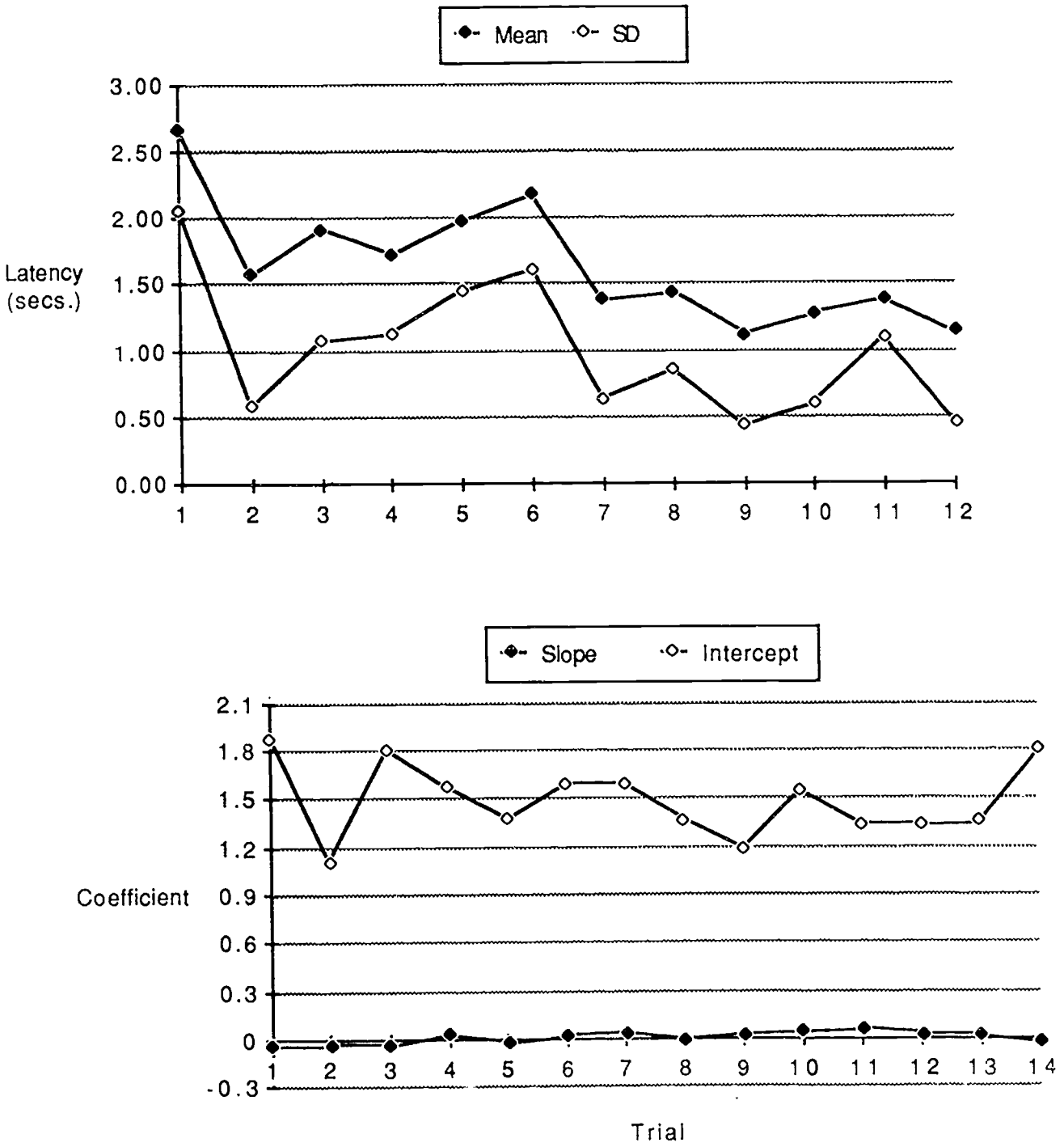


Figure 6. Mean latencies, standard deviations, and regression coefficients for a high school LH student.



AUTOMATH

Basic Math Facts
Extended Practice Program©

Package Contents:

AutoMath User's Manual
A complete Guide to Using the Automath Software Program

AutoMath Teacher's Handbook
A Guide to Effective Use of Math Drill and Practice
Software in the Classroom

AutoMath Master and Preview Disks

Project Developed by

Dorothy S. Semmel

Michael M. Gerber

Melvyn I Semmel

William Lewin, Senior Programmer

The **AutoMath software program**©, the **AutoMath User's Manual**© and the **AutoMath Teacher's Handbook**© were developed under a grant from the Office of Special Education and Rehabilitation Services, U. S. Department of Education, to Instructional Programming Associates, Inc., Santa Barbara, California, Grant Number 026SH70019. The contents of the program do not necessarily represent the policy of the funding agency and no endorsement by the Department of Education is implied.

For further information about the program, write to or call

Instructional Programming Associates (IPA) Inc.
839 Willowglen Road
Santa Barbara, CA 93105 (805) 682-8377

AUTOMATH

BASIC MATH FACTS

Computer-Assisted Practice Program

USER'S MANUAL

AutoMath User's Manual

A COMPLETE GUIDE TO USING THE AUTOMATH SOFTWARE PROGRAM

Field-Test Version
Not for General Distribution
April 1990

TABLE OF CONTENTS

SPEAKING AUTOMATH

AUTOMATH INSTRUCTIONAL ALGORITHMS

OVERVIEW

WHAT IS AUTOMATH?

INTRODUCING THE AUTOMATH PROGRAM

AutoMath is an experimental math drill and practice program designed to aid the student in developing rapid response skills for basic math facts. The ability to rapidly solve basic, single-digit math facts is seen by many researchers as essential to a pupil's effectiveness in solving more complex, multi-digit and math word problems.

The internal programming and the decision-making model incorporated in the **AutoMath** software is based on contemporary research and theory concerning children's development of basic math skills and on the improvement of math performance by low achieving and mildly handicapped students. It is an "intelligent" system that monitors the speed and accuracy of the pupil's performance and adjusts the task as needed for the pupil to acquire rapid and accurate responding.

At the outset, **AutoMath** provides a pretest of the pupil's keyfinding speed to establish a target speed level. This is followed by a quick assessment of the pupil's single-digit problem accuracy which is used to determine which problems are appropriate for practice. The pupil is automatically placed in the correct level for practice and the program continually monitors and adjusts the task in response to the pupil's performance.

A "scoreboard" and graphs showing cumulative accuracy and speed of performance are presented after each 20 problem practice lesson. The program also includes an "evaluation" component that exhaustively tests all single-digit combinations within a given operation. The program automatically places the student in the evaluation component when criterion has been achieved at level "9" problems. However, the teacher may have the student evaluated at any time.

Results of pupil performance are available to the teacher at the problem, lesson, or cumulative lesson levels.

A discussion of the theoretical and empirical basis for the development of **AutoMath** may be found in the **Teacher's Handbook**, which togethe. with the **User's Manual** and the **Master Disk**, comprises the courseware and software for the **AutoMath** package.

The **User's Manual** is designed to make the teacher aware of **AutoMath's** many options in order to aid the student in utilizing the software to its fullest advantages. The first section of the manual, ***Knowing the Basics,*** can be used as a quick reference to all the basic operations of the program. The second section of the manual, ***Program Components,*** provides an expanded guide of detailed information regarding each component of the program. The last section of the manual, ***Speaking AutoMath,*** describes the decision-making algorithms of each major component, and includes a glossary of terms.

Contents of the Program

The **AutoMath** program consists of:

- A User's Manual
- A Teacher's Handbook
- A Master Disk
- A Preview Disk

Note: The teacher should use the **Preview Disk** for practicing and becoming familiar with this program. **DO NOT use the Master Disk** for this purpose. The **Master Disk** is only to be used for making student copies of **AutoMath**, as each student needs his or her own personal **Student Disk**.

Program Features

AutoMath is designed to automatically meet the individualized needs of the student. It provides:

- A **keyboard key-finding speed test** which assesses the student's speed on typing numerals, and establishes the student's target speed of response (automaticity) level.
- The **opportunity to select the math operation** that the student will work on: addition, subtraction, multiplication or division.
- An **assessment of math fact knowledge** for the student, for each of the basic operations.
- **Automatic placement at the appropriate practice level** based on assessment outcomes.
- **Automatic monitoring and analysis** of the student's performance on practice problems.
- **Automatic advance** through all levels of an operation, dependent on the student's performance.
- A **summative evaluation** of the student's mastery of an operation and his or her need for further practice.

Feedback Features

The **AutoMath** program provides the **student** with:

- Immediate feedback on correctness or incorrectness of responses to each problem.
- A **Scoreboard** with summary information on the number and percent correct for each lesson, and the length of time taken to complete that lesson.
- A cumulative graph showing percent correct for each lesson.
- A cumulative graph showing median speed (IKL) of response times for each lesson.

The **AutoMath** program provides the **teacher** with:

- Student's median and fastest speed for the key-finding speed test.
- Placement level for practices, based on assessment.
- Number and percent correct for each lesson.
- List of correct problems, list of incorrect problems, and speed for each.

KNOWING THE BASICS

BEFORE YOU BEGIN

A QUICK GUIDE TO USING AUTOMATH

CAUTIONS

Teacher Roles vs. Student Roles

AutoMath has 6 basic components which should be administered to the student at particular stages of his or her progress, as described in this **User's Manual**. For this reason, it is important that the teacher maintains the responsibility of selecting the student's components from the **Main Menu**. The teacher may or may not wish to supervise the actual practices and testings, but the teacher should be ready to assist the student when component work is completed.

Of the 6 components in **AutoMath**, 4 are designed for student use while 2 are designed for instructor use. Each chapter in the Manual will indicate whether the component is for the instructor or for the student.

Master Disk vs. Preview Disk vs. Student Disk

This package contains only a **Master Disk** and a **Preview Disk**. However, it is necessary that each student have his or her own personal **Student Disk** which is customized for that particular student the first time it is used. The teacher should use the **Master Disk** to make copies of **AutoMath** for each student. The **Preview Disk** is for the teacher's use only, to become comfortable and familiar with the program. **DO NOT** use the **Master Disk** for this purpose.

Directions for Making a Student Disk:

1. Insert the **Master Disk** into disk drive and turn on computer.

2. After about 30 seconds, the "AppleIIGS Program Launcher" will appear, requesting that you select the file you want to open.

3. Select "FINDER" by either:
   a) clicking the mouse on "FINDER" once, then clicking on "OPEN," or
   b) double clicking on "FINDER".

*If you are using a **single-disk drive** system:*

4. Eject the **Master Disk** by pressing the "apple" key (the key with the picture of an apple), in the lower left-hand area of the keyboard, and the **"E"** key, simultaneously. Replace the **Master Disk** with what will be the **Student's Disk**. This disk is called the **Destination Disk**.

5. If the disk has not yet been initialized, the screen will ask if you want to initialize the disk as a ProDOS disk; use the mouse to click "Initialize". The screen will then ask you to name the disk. Type in any name and click "OK".

6. After the initialization has been completed, click and hold the mouse down on the **Master Disk Icon**, dragging the icon to the **Destination Disk Icon**. This begins the copying process of one disk onto the other. Be sure to follow the prompts on switching disks.

7. **Be aware** that if you are using a **Destination Disk** that previously had information on it, that information will be erased when you copy **AutoMath** onto that disk. A message will come up warning that this will happen. If this is OK, click "OK" when the replacement warning appears.

8. You have finished copying **AutoMath** onto your student's **Destination Disk**. Before being able to use it, though, the computer needs to be re-booted (turned off and turned back on).

*If you are using a **double-disk drive** system:*

4. Insert what will be the **Student's Disk** into the empty disk drive. This disk is called the **Destination Disk.**

5. If the disk has not yet been initialized, the screen will ask if you want to initialize the disk as a ProDOS disk; use the mouse to click "Initialize". The screen will then ask you to name the disk. Type in a name and click "OK".

6. After the initialization has been completed, click and hold the mouse down on the **Master Disk Icon**, dragging the icon to the **Destination Disk Icon**. This copies the entire contents of the one disk onto the other.

7. **Be aware** that if you are using a **Destination Disk** that previously had information on it, that information will be erased when you copy **AutoMath** onto that disk. A message will come up warning that this will happen. If this is OK, click "OK" when the replacement warning appears.

8. You have finished copying **AutoMath** onto your student's **Destination Disk**. Before being able to use it, though, the computer needs to be re-booted (turned off and turned back on).

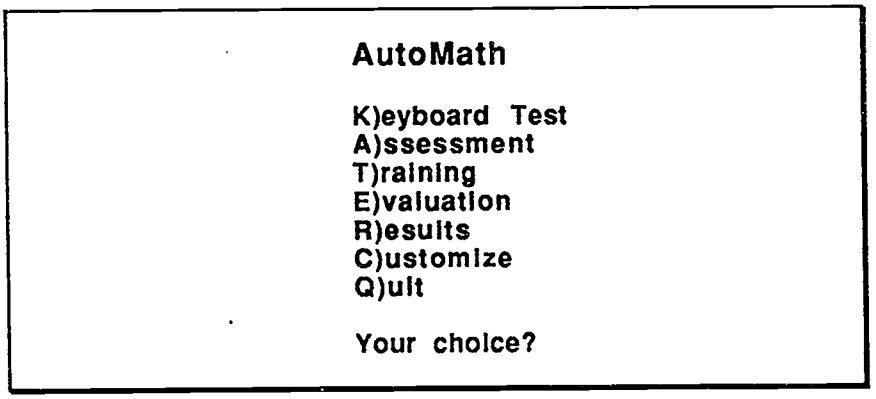
Directions for Getting to the Main Menu

1. Put the **AutoMath** disk into the disk drive and turn on the computer.

2. After about 30 seconds, the "AppleIIGS Program Launcher" will appear, requesting that you select the file you want to open.

3. Select "AUTOMATH" by either:
   a) clicking the mouse on "AUTOMATH" once, then clicking on "OPEN," or
   b) double clicking on "AUTOMATH".

4. When the **Main Menu** appears, it will look like this:



Main Menu Definitions:

K)eyboard Test	Assesses the student's key-finding speed.
A)ssessment	A rapid assessment of the student's accuracy with single-digit math problems for placement in the appropriate practice level.
T)raining	Selects and presents 20 math problems (2 sets of 10) for student's training and practice at a level determined through **Assessment**.
E)valuation	Assesses accuracy and speed of all single digit problems in a given operation.

C)ustomize Used to select operations, change levels or change settings.

Q)uit Permits you to safely exit the program.

Note: At the end of a program component, it is necessary to always return to the **Main Menu** before accessing another component or before **Quitting**. To access the **Main Menu**, either follow screen commands, or press the **"control"** key and **"T"** key, (termed **"control T"**) simultaneously.

Also note that to choose a component from the **Main Menu**, the teacher need only to **press the letter followed by the paren**. For example, to choose **C)ustomize**, press the **"C"** key.

Individualizing a Student Disk

Each disk must have a password that identifies the exclusive student user of that disk. During the **Keyboard Test** component of the program, the student is asked to type his or her name. This name becomes the student's password and identifies the student's disk.

Note: The student's name must be spelled the same way each time in order to run the program.

Selecting the Math Operation

The **Master Disk** is originally set for assessment and training in ADDITION. Therefore, all disks copied from the **Master Disk** are set for assessment and training in ADDITION. If you want the student to work in a different operation, select **"C"** for **Customize** from the **Main Menu**.

When the **Customize** menu appears, select **Settings.** Choose the operation of your choice: either **Addition, Subtraction, Multiplication**, or **Division**. **Quit** to return to the **Main Menu**.

AutoMath's Timing Method

It should be noted that **AutoMath** uses what is called the **IKL** as its method of timing. **IKL** stands for **Initial Keyboard Latency**, and it simply means that whenever **AutoMath** is timing the student, it is measuring the amount of time (to the closest tenth of a second) it takes for the student to enter the **first digit** of the correct answer.

1. Put the **AutoMath *Student Disk*** into the disk drive and turn on the computer.

2. Follow the prompts, as directed above, until reaching the **Main Menu**.

3. To begin the first component, press the **"K"** key for **Keyboard Test**.

4. After the student completes the **Keyboard Test**, the screen tells the student to "PLEASE TELL YOUR TEACHER". At this time, the teacher needs to press **"control T"** to return to the **Main Menu**. Then select **Assessment** from the **Main Menu**, again by pressing the letter followed by the parenthesis.

5. When the **Assessment** is over, the program again asks the student to "PLEASE TELL YOUR TEACHER". Follow instructions above to return to the **Main Menu**.

6. The student is now ready to begin the **Training** component. Select **"T"** from the **Main Menu.**

   - After the student completes 20 practice problems, a TRAINING SCOREBOARD, an ACCURACY GRAPH, and a SPEED GRAPH will appear, followed by the message: "GOOD JOB. PLEASE TELL YOUR TEACHER."

   - The teacher may now remove the disk and turn off the computer, or have the student do a second set of 20 problems by pressing **"control T"** to return to the **Main Menu** and again selecting **Training.**

7. At any time after the **Training** component has been completed, various summaries of student performance can be viewed by selecting **"R"** for **Results** from the **Main Menu** and following the prompts as indicated on the screen.

- The **Preview Disk** is only for the teacher to practice and become familiar with **AutoMath**. **DO NOT** use the **Master Disk** for this purpose.

- Students must work ***only*** on their own personal disks which are customized for them the first time they use the program. Use the **Master Disk** to make **AutoMath** copies for each of the students. ***Never*** allow students to "borrow" other students' disks.

- Each student may choose to work on either the number pad or the keyboard pad. But for reliable results, the student must always be consistent in his or her choice.

- Disks may be removed only after selecting **Quit** from the **Main Menu**.

- **Printer Output** -- If the printer is not "on-line" and/or ready to print, the screen will display a prompt saying "NOT SELECTED," and the program will not continue until you press the SELECT or READY button. The printer will then print out the information requested.

Caution: **Do not select Printer Output if a printer is not attached to the computer and turned on.**

PROGRAM COMPONENTS

KEYBOARD TEST

ASSESSMENT

TRAINING

EVALUATION

RESULTS

CUSTOMIZE

The **Keyboard Test** is for student use. It assesses the student's key-finding speed in order to establish a target speed level for later problem solving.

Preparing the Student

The student should be told tc type each number as it appears on the screen and to press the **"enter"** key after each entry. He or she is to work as quickly as possible without making mistakes. However, if mistakes are made, the student should also be told that mistakes can be corrected by using the **"delete"** key and then typing in the correct answer.

Remember, each student may choose to work on either the number pad or the keyboard pad (although the number pad is recommended). For reliable results, the student must always be consistent in his or her choice.

Before the Keyboard Test Begins

1. Open the program to the **AutoMath Main Menu**.

2. The **Main Menu** asks the teacher to choose one of the program components. Select the **"K"** key for **Keyboard Test.**

3. The "KEYBOARD SPEED" screen will appear. Press any key to continue or press the **"esc"** key in the upper left-hand corner to exit.

4. Before the key-finding speed test can begin, the screen will ask the student to type his or her name. This name customizes the student's disk and becomes the student's password. The password also helps to identify a student's disk and prevents the student from using other students' disks.

Note: The student's name must be spelled the same way each time the program is accessed or the program will not run.

5. Directions will then be given to the student for the "FIND-THE-NUMBER" Game. The screen reads:

> **DIRECTIONS:**
>
> **1) FIND THE NUMBER.**
>
> **2) TYPE THE NUMBER.**
>
> **3) PRESS ENTER.**

6. Directions will be followed by a note on using the **"delete"** key and a reminder to the student to work fast:

> **TO FIX A MISTAKE,**
>
> **PRESS DELETE.**
>
> **(Student's Name),**
>
> **WORK FAST.**

7. These two reminders are followed by a brief tutorial on using the **"delete"** key. The screen will read:

> **TO FIX A MISTAKE,**
>
> **PRESS THE DELETE KEY.**
>
> **TYPE 2**

8. The student is to type the number "2" key. Once the student has done this, the screen will tell the student to "PRESS DELETE.":

```
TO FIX A MISTAKE,
PRESS THE DELETE KEY.

TYPE 2    2
PRESS  DELETE.
```

9. The student should then press the **"delete"** key and follow the screen command to "TYPE 3":

```
TO FIX A MISTAKE,
PRESS THE DELETE KEY.

TYPE 3
```

10. Once the student can input correctly and use the delete key, he or she is ready to begin the **Keyboard Speed Test** by pressing enter.

Beginning the Keyboard Test

The student is given 40 key-finding problems in 4 sets of 10. Between sets, a screen appears telling the student how many numbers he or she has to go:

```
GOOD TYPING.
YOU HAVE 30 NUMBERS TO GO.

PRESS ANY KEY TO CONTINUE.
```

When this message appears, the teacher should be sure the student knows to continue the **Keyboard Speed Test** by pressing any key.

When the game is completed, the message, "GAME OVER. GREAT JOB." will appear followed by a screen that reads:

> **GAME OVER.**
>
> **PLEASE TELL YOUR TEACHER.**

To return to the **Main Menu**, press **"control T"**.

With the **Keyboard Test** completed, the **Assessment** component can begin.

The **Assessment** component is for student use. Its function is to make a quick assessment of a student's math level by presenting the student with only a small sample size of math questions.

Before Assessment Begins

Before beginning the **Assessment** component of the program, the teacher should make sure that the program has been set to the math operation (addition, subtraction, multiplication or division) that the student is to be tested in. Directions for selecting an operation can be found under ***PROGRAM COMPONENTS: Customize*** *Detailed Settings.* If no selection is made, the program will default to addition.

Beginning the Assessment

1. Select **Assessment** from the **Main Menu**.
2. The "ASSESSMENT" screen will appear telling the teacher and student what math operation is in use. Press any key to begin (or **"esc"** to exit).
3. The student should type his or her answer to each problem, working as quickly and accurately as possible, and pressing the **"enter"** key after imputing answers.
4. When the assessment is complete, the screen will show the message:

> **GOOD JOB.**
>
> **PLEASE TELL YOUR TEACHER.**

5. To return to the **Main Menu**, press **"control T"**.

When both the **Keyboard Test** and the **Assessment** components are completed, the student is ready for **Training**.

The **Training** component is for student use. After the student has completed the **Keyboard Test** and the basic math facts **Assessment**, he or she is ready to practice solving math problems. **AutoMath** will automatically place the student into the appropriate difficulty level for practice, based on his or her performance in the **Assessment** component. In **Training**, the student is asked twenty basic math questions in 2 sets of ten.

Note: If the **Keyboard Test** and the **Assessment** components have not been completed, **Training** lessons will begin at **Level 5**.

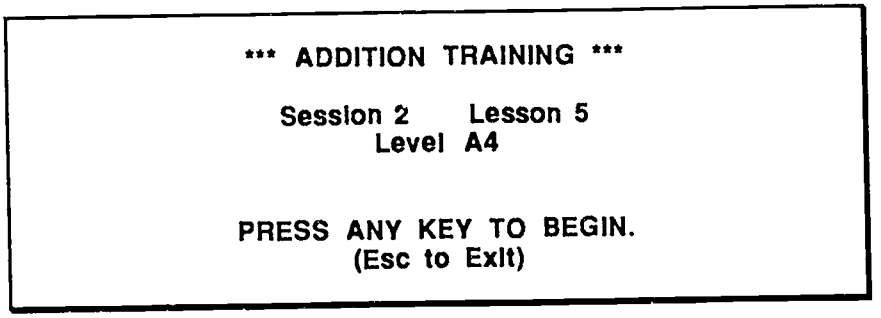
Directions for the Training Component

1. Select **Training** from the **Main Menu**.

2. The first screen that appears will tell the teacher and student:

   a) What math operation is in use. If the teacher wishes to change this operation, press **"esc"** to exit and follow the directions for selecting an operation found under ***PROGRAM COMPONENTS: Customize*** *Detailed Settings.*
   b) What **Session** is in use. **Session** counts the number of **Training** lessons at the present level. Each time the student moves into a new level, **Session** resets to 1.
   c) What **Lesson** is in use. The **Lesson** refers to the total number of **Training** sessions the student has completed.
   d) What **Level** is in use. The **Level** refers to the math operation (A,S,M, or D) and the basic math level (whether the student is working on his or her 3's, 5's, or 9's, etc.) in use.

   The screen might look like this:



This screen indicates:

a) **Addition** is the math operation in use.
b) **Session 2** means that this is the second time that this student has worked at the **Level A4**.
c) **Lesson 5** means that the student has completed 4 **Training Lessons** and is now on **Lesson 5**.
d) **Level A4** means that the student is working on Addition (A) with his or her 4's: 4+3, 4+6, 4+8 etc.

3. When the student is ready to start, press any key to begin (or **"esc"** to exit).

4. Basic math problems will appear on the screen one at a time, depending on the student's level. The student should be encouraged to respond as quickly as possible without making mistakes, always remembering to press **"enter"** after typing his or her answers.

5. There are 3 types of feedback that the student might receive:

   a) If the student answers the question correctly, a message appears reading: "CORRECT!" The next problem comes up automatically.

   b) If the student answers the question incorrectly and wants to change it before pressing **"enter"**, this can be done by pressing the **"delete"** key and changing the answer. If **"enter"** is pressed before a wrong answer is deleted, a message appears reading: "THE CORRECT ANSWER IS 36" (or whatever the correct answer is). The next question comes up automatically.

   c) If the student takes too long to respond to the problem, a message appears reading: "PRESS SPACE BAR TO CONTINUE." The student should be warned to avoid this delay because his or her final time will include this delay. If this message should appear, however, the student should press the space bar to continue. The last question will reappear and after answering, the student will receive feedback and the next question will come up automatically.

6. After the first ten problems are answered, a message appears reading: "PRESS ANY KEY TO CONTINUE." Follow this prompt for the last set of ten problems.

Following Training

1. After each **Training** lesson is completed, a "TRAINING SCOREBOARD" appears, informing the student on:

   a) number correct
   b) percent correct
   c) time it took to complete that lesson

2. After the "TRAINING SCOREBOARD" appears, a "PERCENT CORRECT" bar graph appears. The percentage for the lesson which was just completed is red and labeled "TODAY". When ready to see the next graph, press **"return"**.

3. The next bar graph is the "SPEED GRAPH." This graph shows the student's median speed **(IKL)** for each lesson. The lesson which was just completed is red and labeled "TODAY." When finished with the graph, press **"return"**.

4. The "SPEED GRAPH" is followed by the message: "GOOD JOB. PLEASE TELL YOUR TEACHER."

5. At this point, the teacher can either remove the disk, or press **"control T"** to return to the **Main Menu** for another lesson. **Do not** remove the disk until reaching the **Main Menu** and selecting **Quit**.

The **Evaluation** component is for student use. It is another method of assessing the progress and accomplishments of the student. The component asks the student to answer only twenty problems (2 sets of ten) at a time, taken from all levels (0 - 9). **However**, for the student to be thoroughly and accurately evaluated, it is necessary for the student to repeat the **Evaluation** component until the program tells the student that his or her evaluation has been completed. This means that the student must complete the **Evaluation** component **at least 5 times** (one hundred problems), but possibly **up to ten times** (two hundred problems). Each time the **Evaluation** component's twenty problem set is done the results are saved; when the student again accesses the component, the program incorporates the previous **Evaluation** performances. Again, the program will inform the student of when his or her evaluation has been entirely completed.

When to Use the Evaluation Component

The **Evaluation** component can be used at two different stages:

1. **After Level 9** has been completed through the **Training** component stage, the student will automatically step-up into the **Evaluation** component for his or her evaluation. At this time, the student should go through the component, answering the problems as quickly and accurately as possible. After completing his or her entire **Evaluation** (one-hundred to two-hundred problems), the student will be told whether or not he or she has **"graduated"** from that operation. If the student has **graduated**, the program gives its congratulations; if the student has not, he or she is told to go back to the **Training** component where the evaluation has reassessed the appropriate problem level based on the student's **Evaluation** performance.

2. **Anytime after** the student has worked in either **Keyboard Speed** or **Training**, he or she can be reassessed using all one-hundred single-digit math problems in the matrix (0 + 1, 0 + 2, ... 9 + 9) by completing the 5 to ten step **Evaluation**. **Graduation** is quite improbable if the student has not completed the **Training** component up through **Level 9**; therefore, when the **Evaluation** component is used before **Level 9** has been reached it is basically working as another assessment.

   The purpose of having either the **Keyboard Speed** or the **Training** components done is because the **AutoMath** program needs an **Evaluation Time Criterion Source** for the student's **Evaluation**. An **Evaluation Time Criterion Source** is simply the problem time limit that **AutoMath** uses throughout the program to measure a student's accomplishments. The program's first choice for an **Evaluation Time Criterion Source** is the mean speed plus 2 times the standard deviation established during the student's **Training** component. If the

Training component has not been completed, the program automatically uses the median speed plus 2 times the standard deviation established during the **Keyboard Speed** component.

Again, once the student has completed the entire **Evaluation**, and has not **graduated**, he or she should go back to the **Training** component where the evaluation has reassessed the appropriate problem level based on the student's **Evaluation** performance.

If neither the **Keyboard Speed** test nor the **Training** component have been completed, the program lacks a speed measurement to evaluate the student. In this case, a screen will appear asking the student to "PLEASE COMPLETE THE KEYBOARD TEST BEFORE CONTINUING WITH THE EVALUATION." Again, the student could also complete the **Training** component to fulfill the requirement regarding the **Time Criterion Source**.

Directions for the Evaluation Component

1. Choose **Evaluation** from the **Main Menu**, or if **Training Level 9** has been completed, the student will already have automatically been placed in the **Evaluation** component.

2. If neither the **Keyboard Speed** nor the **Training** component have been completed, a screen will come up stating "PLEASE COMPLETE THE KEYBOARD TEST BEFORE CONTINUING WITH THE EVALUATION." To do this, ...

3. If all prerequisites have been fulfilled, the "EVALUATION" screen will appear telling the teacher and student which math operation is in use, as well as what **Session** of the **Evaluation** component the student is on. The screen indicates how many times the student has finished a twenty-problem set of the evaluation. Remember, to be accurately evaluated, the student must complete all the **Sessions** until the program informs the student that the **Evaluation** is done. This will be take between 5 and ten **Sessions**.

4. When the student is ready to start, press any key to begin (or **"esc"** to exit).

5. Basic math problems will appear on the screen, one at a time, regardless of the student's level. The student should be encouraged to respond as quickly and accurately as possible, always remembering to press **"enter"** after typing his or

6. Possible feedback is as follows:

   a) If the student answers the question correctly, a message appears reading: "CORRECT!" The next question comes up automatically.

   b) If the student answers the question incorrectly and wants to change it before pressing **"enter"**, this can be done by pressing the **"delete"** key and changing the answer. If **"enter"** is pressed before a wrong answer is deleted, a message appears reading: "THE CORRECT ANSWER IS 36" (or whatever the correct answer is). The next question comes up automatically.

7. After the first ten questions are answered, a message appears reading: "PRESS ANY KEY TO CONTINUE." The student should follow this prompt for the last set of ten problems.

8. The second set of ten problems is followed by one of three messages:

   a) "CONGRATULATIONS. YOU ARE FAST IN [OPERATION]. PLEASE TELL YOUR TEACHER." if all **graduation** criteria have been met.
   b) "PLEASE WAIT. I'M CHECKING YOUR WORK."
   "YOU NEED TO PRACTICE ON YOUR [LEVEL]'S. PLEASE TELL YOUR TEACHER." if evaluation is complete but the student did not **graduate**.
   c) "GOOD JOB. PLEASE TELL YOUR TEACHER." if the student's evaluation is not yet complete.

9. The teacher may now press **"control T"** to return to the **Main Menu**.

Note: For a detailed explanation of **graduation** criteria and other algorithm related questions, see ***SPEAKING AUTOMATH: AutoMath Instructional Algorithms***.

Reevaluating

If the student has already been evaluated once and asks to be reevaluated, when the **Evaluation** component is chosen, a special screen will appear reading: "EVALUATION COMPLETE. PLEASE TELL YOUR TEACHER. WOULD YOU LIKE TO REEVALUATE? (Y/N)" The teacher can then decide whether or not to have the student reevaluated.

If the teacher chooses "Y" for yes, another screen appears warning the teacher and student that reevaluating erases the previous **Evaluation** data and gives the teacher the chance to again decide whether or not to continue. Note that if the previous **Evaluation** data has already been reviewed by the teacher through the **Results** component, there really is no need to retain the old data.

If the teacher chooses "N" for no, **AutoMath** will return to the **Main Menu**.

The **Results** component is for instructor use. It allows the teacher to check a student's work and/or progress at any time. In this section, we will first describe the **Results Component Menu** and define generally each of the options; then we will take a more detailed look at each of these options.

Results Component Menu

After choosing **Results** from the **Main Menu**, a screen will come up entitled "RESULT'S FILE on [Student's name] DISK."

On the initial screen, the results of each **Training** component completed by the student is shown, including the DATE, LESSON, OPERATION, and LEVEL.

Also found on the first screen is the **Results Component Menu** with all the options for the **Results** component. The bottom of the screen will look like this:

T)raining	**C)umulative summary**	**D)isplay data**
K)eyboard speeds	**E)valuation**	**eV)aluation summary**
	Q)uit to main menu	

Results Menu Definitions:

T)raining — Displays graphic representations of a student's percent correct and median speed across all **Training** lessons. Provides: Cumulative **Training** results, with option to print; Percent accurate by lesson, with option to print; Median latency by lesson, with option to print; Pool latencies, with option to print.

C)umulative summary — Provides a report of individual **Training** lessons, including date completed, operation type and level, percent correct, and median correct speed, with option to print.

D)isplay Data — Provides data summaries of individual **Training** lessons with option to list each problem, student's answer, and initial latency, with option to print.

K)eyboard speeds — Provides individual **Keyboard Test** results, including median speed and low latency, with option to print.

E)valuation	Lists **Evaluation** files and provides data summaries of individual **Evaluation** sessions, with option to list each problem, student's answer, and initial latency, with option to print.
eV)aluation summary	Provides a summary of **Evaluation** data and present status, including number of sessions completed, total problems evaluated, total **Evaluation** problems correct and total **Evaluation** problems slower than the **Time Criterion Source** allowed, with option to print.
Q)uit to main menu	Brings you back to the **AutoMath Main Menu**.

Results Menu in Detail

Training

Displays graphic representations of the percent correct and of the median speed across all **Training** lessons. Also displayed are the cumulative **Training** results, with option to print. The screen might look like this:

ADDITION Session Results on BILL'S DISK.

Total sessions to date: 3
Total problems to date: 60
Total correct to date: 38
Percent correct to date: 63.33
Current Level: A6
Sessions in current level: 2
Lesson Sequence Transitions: 2
Lowest Latency of Previous Session: 0.43
Variance of Adjusted Session: 0.02

This option also displays data for percent accurate by lesson, median latency by lesson, and pool latencies, all with option to print.

Cumulative Summary

Provides a report of individual **Training** lessons, including: date completed, operation type and level, percent correct, and median correct speed, with option to print. The screen might look like this:

CUMULATIVE SUMMARY REPORT

DATE COMPLETED	TYPE & LEVEL	PERCENT CORRECT	MEDIAN CORRECT SPEED
07/02/89	A7	100.00%	2.5
07/02/89	A7	90.00%	2.7
07/02/89	A7	95.00%	2.3
07/02/89	A7	100.00%	2.2
07/04/89	A7	95.00%	2.2

Display Data

1. Displays data summaries of selected individual **Training** lessons, with option to print. The first screen might look like this:

AUTOMATH -- DISPLAY TRAINING RESULT FILES

INSTRUCTIONAL SET SUMMARY BILL'S DISK.
DATE: 07/02/89 OPERATION/LEVEL: A7
LESSON: 3 SESSION: 3

NUMBER OF PROBLEMS WORKED: 20
MEDIAN SPEED FOR CORRECT PROBLEMS: 0.70

CORRECT : 18 (90.00%)
INCORRECT : 2 (10.00%)

2. Option to list each problem, student's answer, and initial latency, with option to print. Both on the screen and when printed, the problems are grouped by correct and incorrect answers. Correct problems are listed first, then the program asks the teacher to "PRESS ANY KEY TO CONTINUE" in order to see incorrect problems. The screens might look like this:

CORRECT PROBLEMS 07/02/89 LESSON: 3 SESSION: 3

PROBLEM	STUDENT ANSWER	INITIAL LATENCY
3+7	10	2.5
2+7	9	0.7
3+7	10	0.6
0+7	7	0.7
3+7	10	0.6
8+7	15	0.6
1+7	8	0.8
0+7	7	0.8
4+7	11	0.6
2+7	9	0.6
1+7	8	0.7
2+7	9	0.7
4+7	11	0.6
3+7	10	0.7
0+7	7	0.8
4+7	11	0.7
2+7	9	0.9
0+7	7	0.8

INCORRECT PROBLEMS 07/02/89 LESSON:3 SESSION:3

PROBLEM	STUDENT ANSWER	INITIAL LATENCY
2+7	8	0.9
1+7	7	0.8

Keyboard Speed

Displays individual **Keyboard Test** results, including median speed and low latency, with option to print. A screen might look like this:

KEYBOARD SPEED RESULTS on BILL'S DISK.

Median: 1.5
Low Latency: 1.1

67- 1.9	40- 1.8	2- 2.0
45- 1.4	15- 1.4	50- 1.8
66- 1.5	16- 1.7	81- 1.5
3- 1.3	41- 1.5	8- 2.0
42- 1.3	36- 1.4	58- 2.1
66- 2.1	46- 1.2	62- 1.7
54- 1.5	32- 2.0	17- 2.1
4- 2.0	62- 1.3	9- 1.6
36- 1.8	26- 1.7	19- 2.2
41- 1.2	56- 1.6	43- 1.5
63- 2.1	22- 1.5	
0- 1.1	49- 1.2	
69- 1.3	29- 1.2	
12- 2.0	56- 1.4	
2- 1.5	80- 1.3	

Evaluation

Lists **Evaluation** files and displays data summaries of individual **Evaluation** sessions, with option to list each problem, student's answer, and initial latency, with option to print. A data summary might look like this:

DISPLAY EVALUATION RESULTS BY SESSION

DATE: 07/02/89 SESSION: 2
PROBLEMS PRESENTED: 20
NUMBER CORRECT: 6
NUMBER SLOW ONCE: 6
NUMBER SLOW TWICE: 13
EVALUATION TIME CRITERION MEAN: 2.0000000e-2
EVALUATION TIME CRITERION SOURCE: M

Evaluation Summary

Provides a summary of **Evaluation** data and present status, including number of sessions completed, total problems evaluated, total **Evaluation** problems correct and total **Evaluation** problems slower than the **Time Criterion Source** allowed, with option to print.

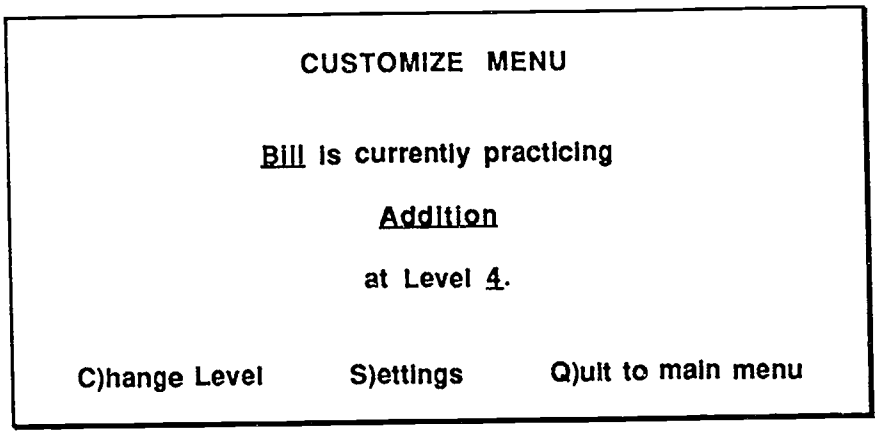
Quit to Main Menu

Brings you back to the **AutoMath Main Menu**.

The **Customize** component is for instructor use.The function of this component is to allow the teacher options in changing lesson levels and/or operations, and in adjusting default **AutoMath** settings. Teachers will want to be aware of each individual option in this component, as the flexibility that **Customize** allows can be extremely useful in terms of unique student needs and class objectives.

The Customize Menu Screen

After choosing **Customize** from the **Main Menu**, the first screen to appear will be the **Customize Component Menu**. The screen might look like this:



This screen identifies a number of things:

1. In the second line, the teacher is told which student this disk is individualized for.

2. In the third line, the teacher is told which operation the student is currently working on.

3. In the fourth line, the teacher is told at what **level** the student is currently working on. Remember that **level** refers to the the common digit in each problem at that **level**. In other words, since the student above is at **Level 4**, he is working on problems such as 4+0, 4+1, 4+2, etc.

4. The last line in the screen above identifies the teacher's **Customize Menu** options.

Customize Component Menu

In the following sections, we will first describe the **Customize Component Menu** and generally define each of the options; then we will take a more detailed look at each of these options.

The menu for the **Customize** component is as follows:

C)hange Level	S)ettings	Q)uit to main menu

Customize Menu Definitions:

C)hange Level	Provides the option of changing the student's problem **level**.
S)ettings	Provides option of changing default settings, including operation.
Q)uit to main menu	Brings you back to the **AutoMath Main Menu**.

Customize Menu in Detail

Change Level

As stated above, the **Change Level** option allows the teacher to manually select the **level** for the student.

To do this, the teacher needs to simply select **"C"** for **Change Level** from the **Customize Component Menu**. When this has been done, a "WARNING" screen will appear, cautioning the teacher that changing **levels** may erase the current **level** data. This is just to say that once **levels** are manually changed, all old data is cleared from the **Results** component. If that data has already been reviewed and the teacher no longer has any use for it, there is really no reason to retain it any longer.

Therefore, if the teacher wishes to change levels, choose **"Y"** for yes. If the teacher does not wish to change levels, choose **"N"** for no to return to the **Customize Component Menu**.

If the teacher chooses to proceed, another screen will appear that will look similar to this:

```
I want Bill to practice

problems at Level _.

(Enter 0 - 9)

Q)uit to Customize Menu
```

By simply entering a number from 0-9, the teacher can manually select the **level** at which the student will practice.

When this has been done, select **"Q"** to return to the **Customize Component Menu.**

Settings

Provides the teacher with the opportunity of changing default settings. The **Settings** screen and the **AutoMath** defaults are as follows:

```
AUTOMATH SETTINGS

A. OPERATION  (A,S,M,D)? A

B. SOUND ON? N
C. NUMBER OF LATENCY TEST NUMBERS (40 MAX)? 40
D. TRAINING & EVALUATION RESPONSE TIME LIMIT? 10.0
E. ASK ADDITIONAL TRAINING LESSON? N
F. SKIP MENU TO TRAIN? N
G. ALLOW ESC IN ASSESS, EVALUATION & TRAIN? N
H. REQUIRE STUDENT NAME FOR TRAINING, ASSESS &
   EVALUATION? N
I. STUDENT NAME: [Disk's Password/Name]
J. REQUIRE SPACE BAR PRIOR TO EACH PROBLEM? N

A) ... J)        Q)uit to Customize menu _
```

To change any of the defaults, enter the letter of that **setting**.

- If it is a **setting** which must be answered with either a **"Y"** for "yes" or a **"N"** for "no," simply choosing the letter of that **setting** changes the default and nothing more needs to be done.

- If it is a **setting** with a number of possible answers, after choosing the appropriate letter, the cursor will move to the present answer where the teacher can input the customization.

Enter **"Q"** when ready to return to the **Customize Component Menu**.

Settings Menu in Detail

A. OPERATION (A,S,M,D)? A

The default for the **Operation Setting** is **Addition**.
This option controls the operation in use for all the **AutoMath** components.
To change the operation that the student is currently working on, enter either **"S"**, **"M"**, or **"D"** for **Subtraction**, **Multiplication**, or **Division**, respectively, after choosing **"A"** at the "AUTOMATH SETTINGS" screen.

B. SOUND ON? N

The default for the **Sound On Setting** is **No**.
This option is preset with the sound turned off so as not to disturb other students in the classroom. With the sound turned on, two short beeps will precede each math problem. This can be used to alert the student that another problem will soon appear on the screen. If beeps are too loud, the sound can be adjusted. Refer to documentation for the computer hardware.
To change the **Sound On Setting**, just enter **"B"** at the "AUTOMATH SETTINGS" screen.

C. NUMBER OF LATENCY TEST ITEMS (40 MAX)? 40

The default for the **Number of Latency Test Items** setting is **40**.
This option controls the number of items in the **Keyboard Test** component. It is preset to 40 to provide an adequate number of test items for establishing a **Median Time Criterion.** For this reason, **changing this option is not recommended**.

D. TRAINING AND EVALUATION RESPONSE TIME LIMIT? 10.0

The default for the **Training and Evaluation Response Time Limit Setting** is **10.0** seconds.
This option allows the teacher to pace the student's work. Specifically, it allows the teacher to set a limit on the number of seconds the student may take in answering **Training and Evaluation** problems before the program automatically advances to the next problem. Note that any unanswered problems are considered incorrect. After the student becomes more familiar with the program, the teacher may wish to decrease the **Training and Evaluation Response Time Limit** so that the student will begin to answer problems more quickly.
To change the **Training and Evaluation Response Time Limit**, enter any number, after choosing **"D"** at the "AUTOMATH SETTINGS" screen.

E. ASK ADDITIONAL TRAINING LESSON? N

The default for the **Additional Training Lesson Setting** is **No**.
When this **setting** is on "yes," after **Training**, the student is automatically asked if he or she would like to work on another lesson. Otherwise, the student is told "GOOD JOB. PLEASE TELL YOUR TEACHER."
To change the **Additional Training Lesson Setting**, just enter **"E"** at the "AUTOMATH SETTINGS" screen.

F. SKIP MENU TO TRAIN? N

The default for the **Skip Menu to Train Setting** is **No**.
The program is preset to begin at the **AutoMath Main Menu**. If this **setting** is changed to "yes," the program begins at the **Training** component.
To change the **Skip Menu To Train Setting**, just enter **"F"** at the "AUTOMATH SETTINGS" screen.

G. ALLOW ESC IN ASSESS, EVALUATION AND TRAIN? N

The default for the **Allow Esc in Assess, Evaluation and Train Setting** is **No**.
The program is preset so that once a student begins either the **Assessment**, the **Evaluation** or the **Training** components, he or she must finish these components. If this **setting** is changed to "yes," a student can press **"esc"** at anytime during any of these three components, and the **Results** data are not saved. The teacher may not want the student to have the option of aborting components once they have begun, therefore **changing this option is not recommended**.

H. REQUIRE STUDENT NAME FOR TRAINING, ASSESS AND EVALUATION? N

The default for the **Require Student Name for Training, Assess and Evaluation Setting** is **No**.
If this **setting** is changed, the student will need to enter his or her name at the beginning of the **Training**, **Assessment**, and the **Evaluation** components, as well as at the beginning of the **Keyboard Test**.
To change the **Require Student Name for Training Setting**, just enter **"H"** at the "AUTOMATH SETTINGS" screen.

I. STUDENT NAME: [Student's Password/Name]

The **Student Name Setting** is initially set when the student types in his or her name at the beginning of the **Keyboard Test**. At that time, the name entered becomes the student's password and identifies the disk.
To change the **Student Name Setting**, enter the new password, after entering **"I"** at the "AUTOMATH SETTINGS" screen. **This new password must be used by the student to access the program.**

J. REQUIRE SPACE BAR PRIOR TO EACH PROBLEM? N

The default on the **Require Space Bar Prior to Each Problem Setting** is **No**.
If this **setting** is changed, the student will need to hit the **space bar** after each problem in order to see the next problem.
To change the **Require Space Bar Prior to Each Problem Setting**, just enter **"J"** at the "AUTOMATH SETTINGS" screen.

Quit to Main Menu

Brings you back to the **AutoMath Main Menu**.

SPEAKING AUTOMATH

AUTOMATH INSTRUCTIONAL ALGORITHMS

GLOSSARY OF TERMS

The algorithms used in the **AutoMath** program allow **AutoMath** to make "intelligent" decisions regarding a student's level placement. All of the program's algorithms are based on factors such as, but not limited to, median, mean and standard deviations of speed and accuracy. The development of these algorithms are based on contemporary research and theory. Since the algorithms used are rather complex, only a brief overview of those algorithms used in the **Assessment**, **Training**, and **Evaluation** components will be presented.

Assessment Algorithms

The **Assessment** component of **AutoMath** assesses a practice level, based on student accuracy and established using only a small sample of math problems. **Assessment** starts by presenting problems beginning at **Level 5**. From there, **AutoMath** looks for at least a 2/3 consecutive correct ratio before testing the next level.

Once **Assessment** has been completed, **AutoMath** internally determines the student's performance at each one of three options:

1. The student has mastered **Level X**;
2. The student has a partial grasp of **Level X**; or
3. The student is not yet prepared for **Level X**.

AutoMath then automatically places the student at the level at which he or she has a partial grasp for practice until mastery has been achieved.

Training Algorithms

AutoMath initially sets a student at a **Training** level determined through the **Assessment** of performance. But once this has been done, the student no longer needs to go through the **Assessment** component again, because the **Training** component automatically judges and places students into new levels (either higher or lower) depending on each individual student's progress. There are several criteria that **AutoMath** uses in evaluating a student's progress and in determining when and if a student moves up or down in level difficulty.

Step-Down Criteria

There are two ways in which a student can step or move down a **Training** level.

1. A student will automatically be moved down a level if three sessions within one level have accuracies below 80%.

2. In any given level, there are ten possible questions (in **Level 1**: 1+0, 1+1, ... 1+9). These questions make up the **available problem pool** for that level. Each question is presented twice during each session. If the same question is missed twice at anytime within a level, that question will be dropped from the **available problem pool**.

   For example, a student is at **Level 5**. At each **Training** session at **Level 5**, the student will be presented all the questions in the **available problem pool** twice. If in the first session, the student provides an incorrect answer for the question '5+6' the first time it is presented, but provides a correct answer the second time it is presented, it remains in the **available problem pool**.

   But, if on the student's next session at **Level 5**, the student again misses '5+6' when it is presented, '5+6' will be dropped from the **available problem pool** and it will not be asked in the following sessions.

   When the number of questions in the **available problem pool** has dropped from the original ten down to five, the student will automatically move down a level.

Step-Up Criteria

There are four criteria for stepping or moving up a **Training** level. All criteria must be met before a student can step-up.

1. The student must have completed a minimum of five sessions within the present level.

2. The percentage correct of the last session must be either higher than the session before, or 100%.

Both the third and fourth criterion depend on whether or not there have yet been any automatic level transitions since the student left the **Assessment** component and moved into the **Training** component.

3. The third criterion deals with the current session's lowest latency, or fastest speed. It also deals with the mean of what is called the **low latency pool**. This is the pool of the lowest latencies (high latencies, or slow times, are weeded out) of all sessions at all levels.

   If there have been no level transitions, **AutoMath** compares the lowest latency of the current session to that of the previous session. The lowest latency of the current session must be lower than the lowest latency of the previous session.

   If there have been level transitions, then the lowest latency of the current session must be lower than the **low latency pool mean**.

4. The fourth criterion is dependent upon the **adjusted session variance** and the **adjusted pool variance**. The **adjusted session variance** is the standard deviation of the lowest session latency, squared. The **adjusted pool variance** is the standard deviation of the lowest pool latencies, squared.

   If there have been no level transitions, then the **adjusted session variance** of the current session must be lower than the **adjusted session variance** of the previous session.

   If there have been level transitions, then the **adjusted session variance** of the current session must be lower than the **adjusted pool variance**.

Evaluation Algorithms

Once a student has completed the **Evaluation**, **AutoMath** automatically "evaluates" that student's progress. A completed **Evaluation** ends by either **graduating** a student because he or she has moved through and mastered each level in a given operation and is ready to move on to another operation; or, if the student is not ready for **graduating**, the **Evaluation** component ends by reassessing the student's appropriate level and automatically reassigning to a new level.

To understand the algoritms used in the **Evaluation** component of **AutoMath**, it is necessary to remember that the **Evaluation** component is centrally concerned with the **time criterion source** (the time limit in which a student should try to answer the questions presented). Also remember that if the **time criterion source** is taken from the student's performance in the **Training** component, then the **source** is equal to the established mean speed plus 2 times the established standard deviation. If the **time criterion source** derives from the student's performance in the **Keyboard Test**, the **source** is equal to the established median speed of latency plus 2 times the established standard deviation.

Graduating

There are four criteria that the student must meet before **AutoMath** considers the student to have **graduated**, ie., completed an operation.

1. All the **Evaluation** questions must be answered. This means that the student must repeat the **Evaluation** component until **AutoMath** tells the student that the evaluation has been completed. This could mean that the student would have to repeat the **Evaluation** component up to ten times, depending on performance, before the **Evaluation** has been completed.

2. The percentage correct of all **Evaluation** questions must be at least 80%.

3. The lowest latency of the **Evaluation** component must be less than the **time criterion source**.

4. The **adjusted evaluation variance** must be faster than the **time criterion source**.

New Level Assignment

If upon completing the **Evaluation** component, the student has not fulfilled each of the above criteria, the student's appropriate level is automatically reassessed and the student is assigned to a new level. To understand how this is accomplished, first note that each question that is presented in the **Evaluation** component is necessarily answered in one of three ways. Either:

1. The student answered the question correctly and within the time limit set by the **time criterion source**, and the answer is therefore termed "quick and correct;"

2. The student answered the question correctly, but not within the time limit, and the answer is therefore termed "slow, but correct;" or

3. The student answered the question incorrectly and therefore time is irrelevant.

To reassign the student to a new level based on performance in the **Evaluation** component, **AutoMath** looks at the **Evaluation** questions that were answered "slow, but correct." **AutoMath** then takes the latencies of these questions and the latencies of their reciprocals and forms a pool of latencies. In other words, if '5+9' was answered "slow, but correct," **AutoMath** places the latency of '5+9' in the latency pool, as well as the latency of the question '9+5'.

These latencies are ranked from fastest to slowest. That ranking is factored with the number of problems at each level that were answered "slow, but correct" and uses that factoring to determine the student's appropriate level.

Adjusted pool variance The standard deviation of the lowest pool latencies, squared.

Adjusted session variance The standard deviation of the lowest session latency, squared.

Assessment Basic component of the **AutoMath** program whose function is to make a quick assessment of a student's math level by presenting the student with only a small sample size of math questions.

Available problem pool A grouping of the problems at each **Training** level that might be presented to the student. Problems are dropped from the ***available problem pool*** if the student misses them twice.

Customize Basic component of the **AutoMath** program that allows the teacher to manually select operations and levels, and to change default settings.

Customize Component Menu The options available in the **Customize** component, including: **Change Level**, **Settings**, and **Quit to Main Menu**.

Evaluation Basic component of the **AutoMath** program that assesses accuracy and speed of all single digit problems in a given operation.

Graduate The status a student reaches, as determined through the **Evaluation** component, when he or she has mastered each problem level in a given operation.

Initial Keyboard Latency (IKL) Timing method of the **AutoMath** program which measures the amount of time elapsed from the presentation of a problem to the moment when the first digit of the correct answer is entered.

Keyboard Test Basic component of the **AutoMath** program which assesses the student's key-finding speed in order to establish a target speed level for later problem solving.

Lesson The twenty problem set of student practice in the **Training** component.

Level The math problem set that the student is working on. If the student is working on problems with the common digit of 4 (4+0, 4+1, ... 4+9), the student if said to be working on **Level 4**.

Low latency pool A grouping of the fastest times of all **Training** sessions at all levels.

Main Menu Provides a list of the basic options or components of the **AutoMath** program, including: **Keyboard Test**, **Assessment**, **Training**, **Evaluation**, **Results**, **Customize**, and **Quit**.

Master Disk A copy of the **AutoMath** program, specifically to be used by the teacher for making individual student copies of the program. The ***Master Disk*** should never be used for any purpose other than copying purposes.

Preview Disk A special teacher's copy of the **AutoMath** program, specifically to be used for teacher's practice in becoming comfortable and familiar with the program.

Results Basic component of the **AutoMath** program that provides results for the **Keyboard Test**, the **Training** component, and the **Evaluation** component.

Results Component Menu The options available in the **Results** component, including: **Training**, **Cumulative Summary**, **Display Data**, **Keyboard Speeds**, **Evaluation**, **Evaluation Summary**, and **Quit to Main Menu**.

Scoreboard A **Training** lesson summary that proceeds each lesson. It informs the student on the number of problems correct, the percent correct, and the amount of time that it took to complete that lesson.

Session Measure used to count the number of lessons at each **Training** level. Each time a student moves into a new level, the ***session*** count resets to one.

Step-Down Criteria The two criterion elements that a student needs to meet before **AutoMath** automatically moves the student down a **Training** level.

Step-Up Criteria The four criterion elements that a student needs to meet before **AutoMath** automatically moves the student up a **Training** level.

Student Disk A copy of the **AutoMath** program made from the ***Master Disk*** for student use.

Time Criterion Source The problem time limit that **AutoMath** uses throughout the program to measure a student's accomplishments. The ***Time Criterion Source*** is either established through the **Training** component, at which time it is equal to the student's mean speed plus two times the established standard deviation, or through the **Keyboard Test**, at which time it is equal to the student's median speed plus two times the established standard deviation.

Training Basic component of the **AutoMath** program which is the core of the training and practice aspect which the program provides. During one **Training** lesson, the student is presented with twenty math problems at a level determined through the **Assessment** component.

AutoMath

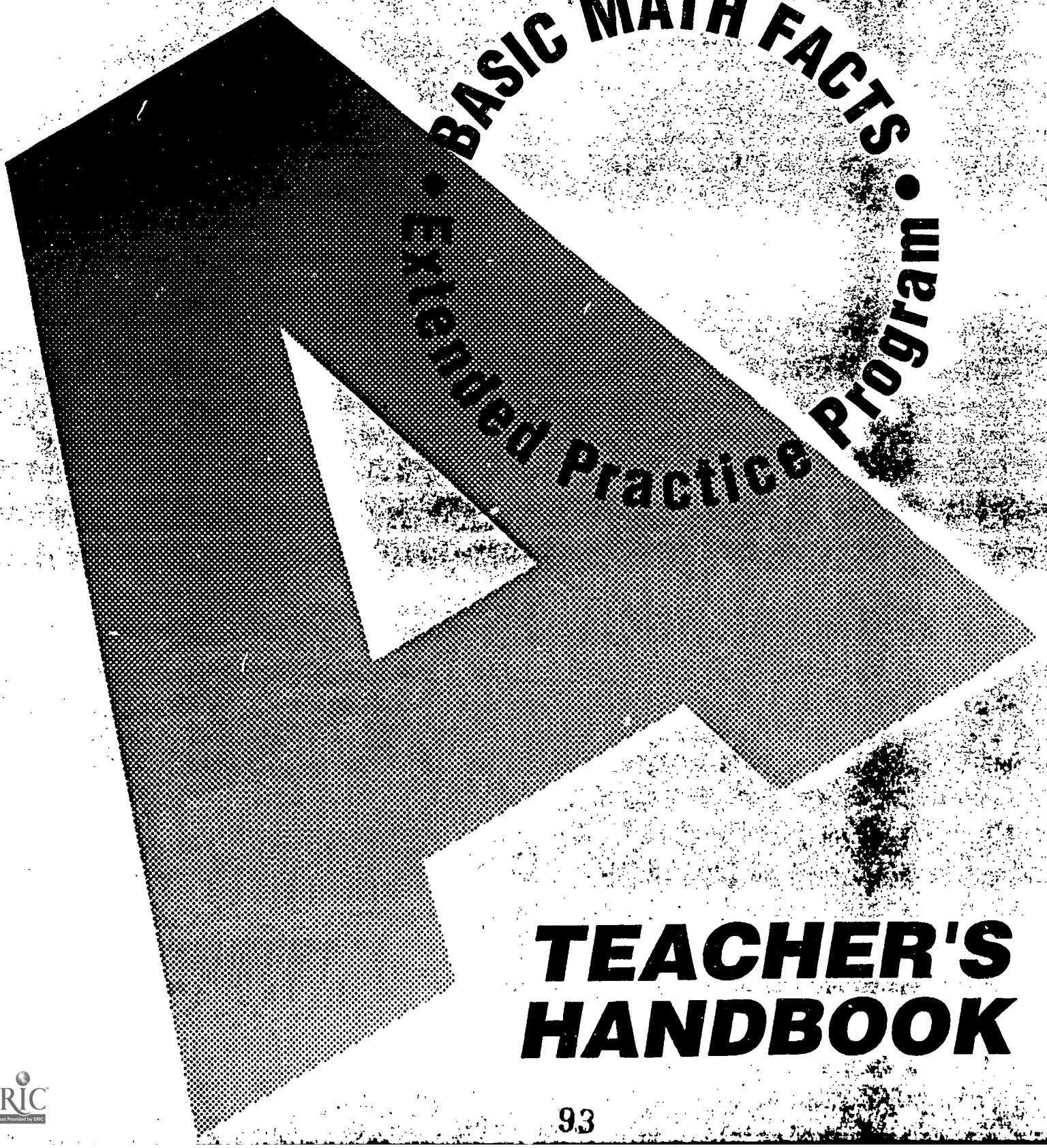
Teacher's Handbook

A GUIDE TO EFFECTIVE USE OF MATH DRILL AND PRACTICE SOFTWARE IN THE CLASSROOM

Field-Test Version
Not for General Distribution
June 1990

AUTOMATH

BASIC MATH FACTS • Extended Practice Program •



TEACHER'S HANDBOOK

TABLE OF CONTENTS

Introduction to Microcomputer-Assisted Development of Math Facts Skills

This handbook has been designed to assist teachers to use microcomputers for effective instruction in the development of pupil's basic math skills. Since the introduction of microcomputers into the classroom several years ago, math drill and practice programs have been the most frequently used type of software with students of all grades and ability levels. Ease of use by both teachers and students has made basic math practice the most popular type of educational software used in classrooms. The relative ease of programming math drill and practice programs has led to a great proliferation of these programs in the software marketplace. Given the widespread use of math programs in the classroom, it is important to know the educational advantages and limitations they present.

Effectiveness of math drill and practice software. In a review of the instructional features of over 60 math drill and practice programs, Semmel & Schnorr (1986) found that only seven of the programs they had located had such important instructional characteristics as timed presentation of problems, teacher or program assessment of pupil entry skill level, or recording and saving pupil performance. Most drill and practice programs were found to be automated workbooks, some with feedback capacity, but generally lacking in any instructional advantage over print practice formats. Thus, in spite of the enormous number of math drill and practice programs both in the public domain and commercially available, few of them meet minimal criteria for providing adequate instruction. Part of the problem was that most drill and practice programs were developed without any theoretical or empirical information on how the microcomputer could be instructionally effective in basic skill improvement. It is only very recently, that research on effective use of microcomputers for basic math instruction has been conducted (Semmel & Schnorr, 1986, Hasselbring & Goins, 1987, Semmel, Gerber & Semmel, 1989, Goldman and Pellegrino, 1986). Results of this research are only now beginning to be reflected in software development.

Another limitation of much of the math software commercially available over the last several years is that teachers find many of the programs difficult to operate. This is particularly true of programs that provide more than simple presentation and feedback. Programs that have added instructional features are often time consuming to learn to operate effectively or require a great deal of disk handling before pupils can use them. The complexity introduced because of the need for multiple disk programs is due to the inherent limitation of the hardware being used in the schools. Most of the microcomputers that schools have acquired have limited memory capacity and therefore need external disks for the microcomputer to handle more complex programs. Ease of use is a fundamental consideration in assessing the effectiveness of any software and it is a feature only found in the most simple and instructionally limited programs. Ideally, a program will have sound instructional features and still be easy to use in the classroom.

How do teachers use math drill and practice software in the classroom? In a four year study of microcomputer effectiveness with mildly handicapped students; Project TEECh (Semmel, M. I., et al., 1986) a series of naturalistic and experimental studies were conducted on teacher and student use of technology. In common with related survey studies, TEECH classroom observation research found that math drill and practice programs were the most frequently used type of software in both regular and special education classrooms. Based on these pervasive findings, Semmel & Schnorr (1987) conducted an observational study of teacher use of math drill and practice software under controlled classroom conditions. They found that pretraining teachers on use of simple, menu driven math drill and practice software, when compared with non-trained teachers, did not make any appreciable difference in how teachers used the programs in the classroom. They further found that teachers did not use the math software to met specific instructional goals for individual students. They were unable to observe any pattern of use that suggested a relationship between computer practice problems selected by the teacher and the on-going (non-computer) math program in the classroom.

In a follow-up study (Semmel, 1988) in which teachers were more extensively trained on the need for preassessment of students to determine which problems pupils should practice, the results were similar to those in the first study. Teachers did not appear to use the knowledge they had gained concerning principles of effective use of math practice software, when assigning their students to computer

practice problems. These findings suggested that teachers were unwilling to allocate the additional time necessary to use the software in a way that was consistent with effective use.

The results of the Project TEECH studies and those of related research (Hasselbring, 1987, Reith & Okolo, 1988), suggest that for a software program to be instructionally effective in the classroom, it must have the instructional decision-making capability that reflects the behavior of an "ideal" teacher, a teacher with sufficient time to pretest and individually select the problems a student will work on to benefit from computer practice. Our research has shown that unless the software is pedagogically capable of an "intelligent" response to pupil performance, and can match the pupil's need for practice, computer practice cannot be expected to play a great role in improving students ability with basic math facts.

The role of practice in the development of basic math facts. The importance of practice in children's mastery of the basic math facts has been the subject of much attention by researchers. Many believe that drill and practice is absolutely essential for strengthening the association between number combinations and thereby assuring rapid retrieval or **automaticity**. Automaticity, or rapid retrieval of facts from memory, is the essential feature of skilled performance. Research indicates that over the course of development, children replace slow counting procedures and thinking strategies with rapid fact retrieval (Ashcraft, 1982; Resnick & Ford, 1981). Many researchers believe that some kind of reproductive process underlies the production of basic facts. That is, through repeated practice the combinations become fixed in memory and thus are available for rapid or automatic retrieval (Ashcraft, 1982).

Other experts suggests that automaticity is the outcome of a reconstructive process involving the pupil's use of stored procedures, rules or principles to quickly construct a range of combinations. This view holds that the reconstructive process is cognitively more economical than reliance on a network of individually stored facts. The reconstructionist position (Baroody, 1985) implies that the essential condition for children's mastery of basic math is their knowledge of the underlying procedures for generating the correct answer. Mastery of the basic math facts therefore implies a conceptual understanding of the fundamental procedures of counting on or down, grouping and separating, as appropriate for the given math operation.

In spite of differing theoretical rationale, both of these views of the development of knowledge of single digit facts imply the need for extended practice for the pupil to attain the automaticity necessary for subsequent development of higher order math skills. Clearly the microcomputer has the potential for making a positive contribution to the pupils mastery of math facts. Through facilitation of practice, the computer is an ideal medium for repeated practice as it is intrinsically motivating for students and appears to hold the attention of students at a high level and over a long period of time. The problem with computer programs, as we have shown, is that the mere presentation of problems is instructionally inadequate. Problems that students practice must be selected to match the individual student's capacity to benefit from practice. Thus, pre-testing or computer-based assessment of pupil math knowledge is essential before students can be assigned to appropriate problems for practice.

This Handbook will discuss techniques for appropriate use of math drill and practice software. The principles of effective use should assist the teacher in evaluating software to use with pupils and in maximizing the benefits of pupil drill and practice time on the microcomputer.

Early math strategies. Children's development of basic mathematical skills follows a sequence in which children initially utilize less efficient counting strategies and later progress to direct retrieval from memory.

Of particular importance during the preschool period is the development of the ability to count. Psychologists have identified several developmental substages that the child progress through in learning how to count. Children also acquire a basic understanding of the "semantics" of addition and subtraction in conjunction with the developmental substages of counting. For example, children first learn to count a number of entities equal to the first addend, then later learn the "cardinality rule," which states that the last counting word produced in the naming of a set of objects is the cardinal number for that set (Fuson, 1982). That is, if a child is told that there are eight objects in a set, he or she knows that if each item is counted, the final counting word produced will be eight.

Employing developmentally acquired knowledge, children determine sums through various counting procedures. One such procedure is referred to as **"counting all".** This method involves determining a sum by counting the total number of entities making up the two addends. Later on, children are observed to use "counting on" methods in which they eventually learn to count out the number of items for the first addend, and then count on from there the number of items equal to the second addend. The age in which this transition occurs has not been established (Case, 1982).

Gradual acquisition of direct retrieval. Children progress in the development of basic fact computation from counting strategies to eventual **automatization** of response. This automatization or "habituation" stage is described by cognitive psychologists as rapid responding with a concomitant absence of identifiable intermediary thought processes, in other words, direct retrieval.

In relation to the acquisition of basic math facts, automaticity is considered an important psychological process. Broverman (1978) defined automatization as the..

"tendency for repetitive routine aspects of behavior to become so overlearned that a minimum of effort and attention is necessary for rapid efficient execution".

In relation to mathematics, automaticity may be defined as the "rapid, effortlessness, errorless direct retrieval of basic facts" (Hasselbring, et al., 1988). Automatic fact retrieval ability is critically important, as it allows limited attentional resources to be used for other components of task performance (i.e. keeping track of place value information).

Subtraction strategies. The issues raised in studies of children's understanding of addition are equally applicable to subtraction fact problems. Preschoolers and children in the early primary grades solve subtraction problems with counting procedures (Starkey & Gelman, 1985; Pellegrino & Goldman, 1987). Just as children use different counting procedures to solve addition problems, they also use a variety of counting procedures to compute subtraction problems.

One of the early attempts to describe counting algorithms used by children to solve subtraction problems was conducted by Ilg and Ames (1943). They outlined two procedures used by young children. One method, used chiefly by 5 year olds, involved counting from the number one up to the cardinal value of the minuend and then counting backwards for a number of steps equal to the cardinal value of the subtrahend. The 6 and 7 year-olds were observed utilizing a decrementing procedure, whereby subtraction problems were solved by counting backwards from the minuend for a number of steps equal to the cardinal value of the subtrahend. Also observed was an incrementing method in which children count on from the subtrahend until the minuend is reached. Depending on the specific problem, one method may be not only easier to use, but more efficient. However, when the opposite situation exists; i.e., there is a small difference between the minuend and subtrahend (e.g., 8-6), incrementing procedures are most useful (Pellegrino & Goldman, 1987).

Multiplication and division strategies. There is as yet, little empirical information regarding the strategies used by children to compute multiplication and division facts. However, the research that is available indicates that diverse strategies are used to compute these problems. Among these are repeated addition or subtraction

(depending on the operation), reference to related problems, and direct retrieval. It appears that in multiplication, the greater the size of the multiplicand and the product, the greater the problem difficulty. However, problems with 5 as a multiplicand and ties (e.g., 7 x 7) are found to be easier than the size of their products would suggest (Campbell & Graham, 1985).

Developing skill in basic operations. The theories of expertise for the remaining basic operations are similar to the theory for addition. They each outline a process in which children initially gather procedural knowledge, followed by the acquisition of declarative knowledge of math facts. The developmental processes occur over a period of many years and will vary dependent upon the amount of experience and the learning characteristics of a given child. Therefore, difficulties with basic facts may be attributed to a weak declarative knowledge base leading to use of counting procedures to compute facts. Further, the counting procedures used for doing such computations are often slow and error prone.

Strategies used by children with learning problems. The research on basic addition and subtraction performance suggests that children with mathematics difficulties often compute, rather than directly retrieve, answers to problems. In one study, learning disabled and nonhandicapped third and sixth graders were tested on twelve basic facts (Fleishner et al.., 1980). The learning disabled students were found to rely more on reconstructive counting strategies than the non-LD students who tended to rely on direct retrieval. In another study, a group of math disabled fourth graders were compared to normally achieving third and fourth graders (Russell & Ginsburg, 1984). They observed particular difficulties in retrieving addition facts by the math disabled students with the children performing at a level below the normal third graders. Svenson and Broquist (1975) have also reported results indicating that fifth grade children with low mathematics achievement are particularly slow at answering simple addition problems. Although researchers have not yet studied multiplication and division, we may assume that the same progression from initially calculating answers to later directly retrieving answers applies also to these math skill areas.

The role of practice in basic facts skill acquisition. The cognitive developmental accounts of the acquisition of basic facts skills provide a useful framework for teachers to use in teaching remedial and learning disabled students to master the basic math

facts. From an instructional perspective, the need for extensive practice is essential. When children have acquired a counting or decrementing strategy and can demonstrate that they can apply it, however slowly, then practice typically serves to strengthen such knowledge. Practice is basic to explaining the acquisition of expertise in simple addition and subtraction.

Using the computer for basic skill development. The microcomputer is an ideal medium for extended practice sessions, especially in basic skills area of mathematics. The microcomputer is capable of providing instructional activities that consistently employ features associated with good instruction. A good software program can provide clear objectives, rapid feedback, good pacing, knowledge of results, and matching of task to learner needs. Microcomputers have also been found to command very high on task behavior from learning handicapped students. The microcomputers' potential for effective instruction, taken together with evidence that one of the primary uses of microcomputers with special needs children is drill and practice in basic math (Semmel, Goldman, Gerber, Cosden & Semmel, 1985), makes it essential that teachers understand how to optimize the use of microcomputer technology in the classroom.

The importance of basic skill development in complex problem solving. The need for the development of a strong foundation in the basic skill areas of mathematics and reading for the learning handicapped cannot be overemphasized. While MH pupils demonstrate a knowledge of **informal** math skills and concepts that are not too different than that of non-math handicapped learners, they are somewhat less efficient (Allardice & Ginsburg, 1983). Differences in math performance of MH pupils are however, most marked in the areas of written calculation and in memory for addition facts (Russell & Ginsburg, 1981). These deficits in the pupil's knowledge of basic facts (single digit problems), significantly interferes with performance on more complex tasks such as carrying, borrowing, self-monitoring and control activities.

Many teachers and parents are satisfied when learning handicapped children are able to figure answers to basic math facts by using primitive counting procedures. Yet researchers who study the ways in which individuals process information believe that when basic skills are overlearned, less attention is required for their execution and they become automatic. This idea is especially significant since all humans have a limited capacity for information processing. If a large portion of the capacity is needed to compute

basic skills, the less capacity will remain for dealing with higher-order concepts and problem solving. It appears that success in higher-order skills is directly related to the degree of efficiency with which lower-order skills are performed. Initial development of procedures for solving basic math fact problems, followed by declarative knowledge and then rapid retrieval of basic math facts, appear to be necessary in order for pupils to free up the attentional resources needed to compute complex math problems.

Why Use Drill and Practice Programs?

Major concepts in this chapter...

Drill and Practice programs are used to practice previously taught material: *They are never used to teach new material; They are used to to follow up previous instruction*

Pupils should practice only at high levels of correct responding: *A correct rate of at least 90% must be maintained over 3 consecutive practice sessions*

The goal of extended practice is to increase the pupils' speed in responding to problems

The primary purpose of drill and practice is to provide students with an opportunity to practice **previously** taught material. Drill and practice exercises are **never** used to teach new material. Instead, they are used to reinforce material previously taught by the teacher and as an aid to rapid retrieval of the answer.

Drill and practice content areas. What content areas are suited for drill and practice software? Researchers have found positive effects of drill and practice for basic math facts, vocabulary and spelling performance. While the present program focuses on math drill and practice, the underlying principles of effective drill and practice appear to apply for vocabulary and spelling as well (Gerber, 1986; Goldman & Mertz, 1986).

Conditions for effective use of drill and practice programs. To work effectively on microcomputer based drill and practice, children must have sufficient knowledge to independently solve problems accurately prior to practicing them on the microcomputer. Children lacking accuracy would benefit from direct instructional assistance and error correction rather than simple drill and practice with feedback. A minimum correct rate of 90% over at least three consecutive practice sessions is recommended prior to implementation of software drill and practice programs that stress rapid responding. If the software permits, it is desirable to delete

from practice all problems which student cannot solve correctly. Thus student correct rate would be at the 95-100% level with only random errors occurring.

Response speed. As has been discussed previously, mildly handicapped (MH) pupils often use counting procedures to compute basic facts. Research has indicated that the use of counting procedures are inefficient in the solution of basic facts problems and interfere with performance on higher-order problems such as multiple digit computation or word-problems. With extensive practice, the student should experience a transition from reliance on counting procedures to direct retrieval of basic facts.

Drill and practice software is most effective in the overlearning phase of learning. That is, effective drill and practice should help the student develop fast and efficient direct retrieval processes, ie, **automaticity**. Fast and efficient processing can be expected only after numerous practice sessions of the same basic fact. To develop automaticity, high levels of accurate responding must be maintained over a protracted period of time. Infrequent, massed practice sessions are not as effective as short, daily practice periods.

The fastest speeds of retrieval of basic math facts, when assessed by keyboard input into microcomputers, are likely to be between 1 second and 1.5 seconds (need citation on speed of retrieval here). Facts that are answered that quickly are "automatized" and are unlikely to show increases in speed through practice. However, there is great individual variation in speed of responding and the 1. to 1.5 second figure represents an average range for non-handicapped students' response to single digit addition problems (citation) Speed of retrieval is an important measure in determining which problems the student should work on. For example, if we use normative criteria and the student retrieves the facts within 1 second to 1.5 seconds, we may assume the student has already "automatized" the fact. If the student responds at a rate slower than 1.5 seconds, it is probable the student has not yet "automatized" the given fact and therefore would benefit from further practice.

Ideally, a program should be capable of determining a target level that is appropriate for the individual student. A software program that accounts for individual differences in response speed would be preferable to one that employs a fixed or arbitrary standard such as the one described above. The need for individualization in the assessment of speed of response and in setting a criterion of automaticity is apparent where the student has

a physical disability that impedes typing. But the need for an individualized automaticity criterion for students with learning problems is also most important since the criterion for the achievement of automaticity must be a close approximation of the fastest response speed that the student is capable of producing motorically.

One method of determining when a given pupil has achieved automaticity is by measuring the pupil's average response speed on a simple keyfinding task. The pupil's response speed on a keyfinding task, which essentially requires a search and motor response, could then be used as a target rate for the achievement of automaticity. The pupil's actual response speed on a basic math problem would consist of the time necessary for retrieval of the answer, plus the time needed for the keyfinding and motor response.

Measure of response speed. A useful measure of pupil's speed of response is the **"initial key stroke latency."** This term refers to the time it takes the pupil to make the first motor response (key stroke) after a problem is shown on the monitor.

An alternative measure of pupil speed of response is the total time elapsed after the problem appears on the monitor. This includes the time it takes a student to input a second or third number, if the correct answer requires more than one digit. It also includes the time to strike the return key. Which ever measure is used, it is essential that a math drill and practice program being considered for classroom adoption have a reliable response speed monitoring capability.

Evaluation and Selection of Drill and Practice Software

Educational software is evaluated to determine its instructional appropriateness and effectiveness. A sound program will have many features that function as if a teacher were present and providing direct, individual instruction. Some important criteria, developed by Schnorr & Semmel (1989), for evaluating basic math drill and practice programs are described below.

Effective drill and practice software should contain the following features or have a have a menu-driven management format that allows teachers to:

- •select specific lessons appropriate for students to practice.
- •create specialized lessons to supplement the preprogrammed lessons.
- •change problem sets by decreasing or increasing the number of problems per lesson.
- •turn the sound system on or off.
- •control the tempo of problem presentation
- •control the student record keeping system.
- •summarizes the students' progress on completed lessons
- •Records problems correct and incorrect
- •records speed of response for each problem
- •review the students' progress on completed lessons.

It is highly desirable for the software to have a data collection file that collects and records information on the students' performance. The resulting record can be most valuable in for instructional assessment and planning. For example, access to pupil

performance data is provides that basis for conducting an error analysis. Programs that provide only global scores or percent correct figures cannot be used in this way. Analysis of pupil's responses to specific problems as recorded by the program, facilitates identification of specific number combinations causing difficulty. Thus, it is most useful for a program to record pupil performance (correctness and speed) at the problem level and also aggregated at the lesson level. Tracking of pupil progress lesson by lesson provides additional information concerning the results of long term practice.

The need for software that measures response speed. Recent research has indicated that the most valuable feature of drill and practice is its ability to develop automaticity. Therefore, programs which measure the pupil's speed of response (i.e., the time it takes a pupil to respond after the problem appears on the screen), and record the response speeds for each problem are highly desirable for assessing the effect of practice in increasing the pupil's speed of response.

Game formats for drill and practice software. Drill and practice software often is presented as an arcade type game because some software developers and teachers feel the competition of the game format and the graphic presentations are highly motivating. These game characteristics, however, may actually be detrimental to skill improvement. The game format is especially detrimental for special needs students who may have difficulty selecting and attending to task relevant information (Christensen & Gerber, 1986). In addition, the game format may be detrimental if the goals and strategies for playing the game conflict with the strategies for completing the academic task.

If teachers select software carefully, it is possible to find game format software that is as effective as software that are without game and graphics features. For the game format software to be as effective, however, the strategies for winning the game must match the strategies for completing the academic task (Goldman & Mertz, 1986).

Many programs include nonessential visual reinforcers like slowly drawn happy faces and other cartoon figures. In some instances, these stimuli require more running time than the drill and practice part of the program. As children's attention is drawn to these stimuli, they effectively compete with the math problems for the child's attention. While observation studies of classroom

computer use have found extremely high levels of pupil on-task behavior across a range of software (Cosden et al.. (1987), indicating yet again that the microcomputer commands a high level of attention from pupils, the game format may be an unnecessary distraction, especially for slower learners. It is therefore preferable that drill and practice software has a minimum of visual and auditory distractors.

Microcomputer-Provided Feedback on Individual Items Effective drill and practice software should provide immediate feedback to the student on correctness of each response. This feedback may include: (1) information indicating whether the response was correct or incorrect, and either (2) the correct answer, or (3) the correct answer and an explanation or illustration of the concept. A tutorial option is desirable for direct instruction purposes, when the objective of instruction is reteaching previously learned materials. While it is not useful in the development of automaticity, it may a valuable adjunct to early instruction.

It is important to note that recent theories about the acquisition of basic skills, especially basic math facts, suggest that practicing problems with incorrect responses strengthens the wrong associations that then compete with the correct answer. The program should never have the student practicing, ie, inputing incorrect responses. It is extremely important that students concentrate their efforts on increasing the speed and accuracy of responses to basic facts they have already learned.

Types of immediate feedback. It is not enough to simply select software that provides immediate feedback. The type of immediate feedback given on correct or incorrect answers is a very important feature of a practice program. Feedback options frequently found are (1) information that the response was correct, in the form of a symbol (e.g., check mark or smile face) or verbal message, (eg, "correct"," great", etc), (2) information that the response was incorrect, with similar symbolic or verbal messages as for the correct response. In some programs, the correct answer may be displayed along with the informational feedback or may stand alone.

Some software packages feature bells and whistles as feedback for incorrect answers. This can create at least two negative conditions: (1) The sound effects draw attention to the student and in effect "punish" the student for incorrect responses, (2) the students may prefer the consequences for incorrect responses more

than the feedback for correct responses. Therefore, the software may unintentionally reward the student for responding incorrectly rather than correctly.

Tempo of presentation. An important consideration in the selection of software is the time lapse between the pupils' last response and the presentation of the next practice problem. For younger children, slow learning or handicapped children, it is important that the speed of problem presentation be amenable to regulation by the teacher. These students would benefit from a slower tempo at least at the outset of practice. Students who have improved their speed of response to a given set of problems can be cued to respond even faster is the presentation tempo is increased.

Microcomputer Summaries of Pupil Performance: Drill and practice software should have the capability of storing individual student records over successive practice sessions. These records provide valuable information about student progress over the course of extended practice sessions. Examining trends in accuracy and in response speed provide teachers with information critical for a variety of instructional decisions. For example, cumulative performance data can assist in the decision to continue or terminate practice, to determine the need for other instructional interventions.

An effective drill and practice software program will at a minimum provide summaries on: (1) Speed of response (2) Number of correct responses (3) Percent of correct responses. These summaries should be available for each practice session and as a cumulative record of performance over all practice sessions. It may also be instructionally useful to have a record of the pupils' performance on each problem. Problem level information can be used for conducting an analysis of the pupil's errors to determine if there are systematic errors attributable to a particular integer or integers.

Substantive feedback should be provided to the student at the end of each lesson. Ideally, the software program should generate a summary on pupil performance that is simple enough for the pupil to understand such as a graph or easy to read "score card". If the program does not have a simple pupil performance summary feature, it is still important for the pupil to know the outcome of repeated practice. This may be done by having the student read the lesson summary and transcribe the outcomes on to a graph or other tally record. If the student cannot do this, then the teacher may provide a written record drawn from the summary data.

It is desirable for students to keep their own handwritten or computer printout record of their performance because it provides a useful self-monitoring function. An alternative technique would be for the teacher to prepare a graph or other tally to share with the student.

Although the microcomputer programs are expected to provide instruction and practice without the active presence of the teacher, few programs have the capacity to do so effectively. As classroom microcomputers become more powerful, the technology may indeed fulfill its promise and provide "intelligent" instruction that adjusts instruction to meet individually assessed pupil needs. In the meantime, microcomputers have an important role to play in instruction. Even at the current state of the art, computers can be a valuable adjunct in an instructional program, if they are used effectively. Guidance for principles of effective use comes from studies of effective teaching in non-microcomputer environments and from studies involving direct observation of microcomputer use in the classroom. This section of the Handbook will review the strategies for maximizing the usefulness of math drill and practice programs in the classroom, including, time allocation and scheduling, assessment, direct instruction and monitoring.

Microcomputer Use Time and Scheduling

Drill and practice software is probably most effective when each student works on the program for at least five minutes per day. If daily session cannot be scheduled, a minimum of 3 times a week for 5 to 10 minutes per day should be provided. Longer lessons once a week are not as effective as more frequent short ones. Math drill and practice programs can also be scheduled according to the number of problems to be practiced in one session at the microcomputer. Generally, sufficient time should be allocated for the student to complete a minimum of two sets of twenty problems. This would normally take no more than 5 minutes to complete, including disk loading time.

It is important that teachers regularly and consistently assign students to the microcomputers, even if it is only for 5 minutes per day in one skill area. Infrequent practice of basic skills results in negligible gains. and practicing basic math facts once per week leads to virtually no performance improvements (Goldman, et al.,1985,1986).

Teaching Strategies: Assessing Students' Instructional Levels

Key Concepts in this section...

•If the software has a pretest, it should be used to assess the appropriate practice level for each student.

*•If the software does **not** have a pretest, use informal, paper & pencil assessment to establish the appropriate practice level.*

•Criteria for appropriate assignment of practice problems:

85 to 90% accuracy on a given set of problems
An average of over 1.5 seconds to respond

The essential prerequisite to effective use of drill and practice programs is assessment of a student's performance. This is the only instructionally valid basis for the prescription of practice problems that are appropriate for the student's individual needs. There are three major variables in assessing a pupil's need. These are:

(1) the pupil's accuracy in responding to single digit problems of a given type and level

(2) pupil's median speed of response to set of problems of a given type and level

(3) the pupil's procedural knowledge of how to solve a given problem

Pretesting to determine pupils' instructional level. Teachers have several options available for assessing a student's instructional level and determining an appropriate initial software assignment: (1) If the software has a pretest, the results should be used to determine appropriate initial practice levels for each student (2) If the drill and practice program does **not** provide a pretest, the program itself could be used to informally determine an appropriate level to initially place the student. (3) an untimed paper and pencil test of basic math problems of a given type may be administered.and problems which pupils respond to **correctly** are then selected for

initial computer practice. (4) A final, but less accurate option would be for the teacher to estimate students' instructional levels based on current placement in the math textbook or workbook.

The concept of problem type and instructional level The effectiveness of extended practice of math facts is based on the assumption that the facts have been previously learned and that through practice the pupil's speed of response will increase to the point of automaticity. To systematically assess and practice, the teacher needs to measure the pupil's performance on one operation at a time. Mastery in basic math usually proceeds hierarchically from addition to subtraction, multiplication and division. Within each of the operations, it is assumed that pupil mastery is related to the size of the number in the addend, subtrahend, etc. That is, the difficulty in solving a given problem increases with the size of the number. The practice level is that number within a given operation which the pupil solves correctly most of the time on pretesting.

Because there is evidence that problem difficulty is related to the size of the number, it is assumed that the pupil can solve all the problems both at and below the assessed practice level. In practice however, the difficulty hierarchy does not hold up for all operations.. For example, students frequently find multiplying by 9 less difficult than multiplying by 6 , 7 or 8. But the number size /difficulty relationship is fairly reliable in addition.

Assessing response speed. Response speed is the second major assessment criterion for determining at which level a student should be working. As discussed elsewhere in the Handbook, the fastest speeds of retrieval of basic math facts, as measured by microcomputer keyboard input, are likely to be between 1 and 1.5 seconds. Facts that are answered that quickly are said to "automatized" and are unlikely to show increases in speed through additional practice.

Assessing procedural knowledge. The third major assessment criteria in support of effective use of drill and practice is determination of pupils' procedural knowledge of a given math operation. It is important to know why students missed problems during assessment. Students are often very slow or miss problems because they use primitive counting strategies or even faulty strategies for obtaining an answer. By examining problems that pupils miss or are extremely slow to respond to, the teacher can determine which problems need further direct instruction before they are practiced for automaticity. Students should not be

practicing problems until they know how to obtain the answer in an efficient manner.and only through pre-assessment can the pupil's stage of procedural knowledge be evaluated.

One method by which students' underlying knowledge may be examined is through dynamic assessment. As described in the section on dynamic assessment, a series of prompts increasing in level of difficulty are presented to the pupil to determine the Zone of Proximal Development (ie, the concept which requires direct instruction from a adult or other model in order for the pupil to master the problem) The prompts in math facts acquisition proceed from a demonstration of the manipulation of concrete objects through to representation of the problem in mathematical form.

The prompting formats used should be based on a concept analysis and on the norms for child development of each of the four basic operations. Testing moves backward from the more abstract representation of the problem through prompts that are increasingly concrete and duplicate the counting stages of informal math development. The dynamic testing should provide useful information about the pupil's present level of math concept development and more specifically, teachers would have a more precise information about the type of instruction necessary to develop the pupil's level of procedural knowledge.

Teaching Strategies: Presentation of Software

Major Concepts in this section:

•Explicitly inform students of the purpose of the software assignment

•Explain the relationship between the computer program and the noncomputer math objectives.

•Inform the student about individual objectives to be met in using the program, i.e., increase speed on answering addition problems.

•Provide examples to the students on how to use the program and demonstrate it for the student.

•Point out any idiosyncrasies or unique characteristics of the program, i.e., format of problems on screen, symbols used other display characteristics, or control features.

•Provide "hands on" practice trials until the student can use the software appropriately.

Direct instruction in use of program. When teachers assign student to drill and practice software, they should begin by orally presenting the software to the students. Initially, the teachers should explain the purpose of the software assignment and the relationship between it and the noncomputer math objectives. In addition to this overall explanation, teachers should set goals for individual student performance, i.e., increase speed on answering addition problems. Next, teachers should give students examples of how to use the software. Once the students have a general understanding, teachers should actually demonstrate the software for the students to show idiosyncrasies and characteristics unique to the program. For example, teachers should point out display characteristics such as the format of problems on the screen, and symbols used, plus the control features.

Teaching Strategies: Monitoring Student Performance While the Pupil Is On the Computer

---

Major Concepts in this section

- *Circulate periodically to encourage on task behavior.*
- *Be available to respond to student requests for assistance.*

Monitoring while pupil is at the computer. In classroom observation studies, Semmel, D.S. et al.. (1986) and Cosden et al.. (1987) found that teachers monitored student performance no more than 10% of the time that students were working at the microcomputer.. Although some software programs have sophisticated record keeping capabilities, teachers still need to maintain an active presence in the microcomputer environment. Teachers need to monitor student performance on the microcomputer on a schedule similar to their schedule for monitoring students' seatwork or workbook exercises so that appropriate adjustments can be made when (1) the student reaches mastery, or (2) the student fails to achieve as expected. This is critical because the student must be on-task and engaged in rapid and accurate responding for the practice sessions to be effective.

The microcomputer cannot distinguish between students who take a long time to answer because they are working out the problem, and students who take a long time to answer because they are not seriously engaged in the task. The teacher however, can make the important distinctions. Teachers also monitor when they provide appropriate feedback to the student by answering a question, guiding a student back to the assigned task, or praising the student for a job done well.

Teaching Strategies: Monitoring Student Performance After Practice

Major Concepts in this section:

•Set up a schedule for periodic monitoring of student performance records.

•Ideally, examine records after each session, or at minimum once per week.

•Assess student performance for:increases in the percent correct and for increases in the speed of response.

•Set up a method for communicating performance results to the pupil.

Checking for student improvement. Student progress should be monitored to know when the student has mastered a skill and is ready for more challenging assignments, or to know that the material is too difficult and that the assignment should be revised.

Optimally, monitoring should occur once per week. Student improvement can be demonstrated through (1) increases in percent correct responses and (2) increases in the speed of responding, i.e., shorter latencies. A latency at or near 1.5 seconds would be evidence of mastery or of the development of automaticity.

Failure to improve. A strategy should be developed to monitor for student failure to improve. If student performance has dropped below the criteria after 3 to 5 sessions, the teacher should take action. The first step is to conduct an error analysis. If the error analysis shows consistent errors, the instructor should reteach those particular problems. If the error analysis reveals random errors, the student should be remotivated.

If the student has still not improved after 6 to 10 consecutive sessions, An error analysis. should be conducted If the error analysis shows consistent errors, the missed problems should be retaught. If the error analysis reveals that the student continues to make random errors, however, the teacher should either (1) remotivate the student, (2) drop the assignment to a lower level, (3) change the

assignment to a new problem set, or (4) change from the drill and practice software assignment and use other methods and media.

Communicating performance data to pupil. The information that the teacher obtains in monitoring pupil performance can also be used to inform the student of progress in meeting math practice goals. It was suggested elsewhere in this program that when the software program is presented to the pupil for the first time, the instructor should tell the student the specific individual goals for speed and accuracy.

It is good practice, therefore, to follow-up on the goals, by providing the student with knowledge of the results of the math practice. This should be done at least once a week, or more often if convenient. The feedback can be done in many ways, from simply telling the student the practice outcomes to setting up creative graphs or charts on which the teacher or the pupil may record progress.

A teacher devised record system could chart progress in pupil accuracy, number of problems completed and average speed of performance.

References

Becker, H.J. (1984) School uses of microcomputers: Reports from a national survey. Baltimore, MD: John Hopkins University, Center for Social Organization of Schools.

Christensen, C. A., & Gerber, M.M. (1986, March). Effects game format in computerized drill and practice on development of automaticity in single digit addition for learning disabled students. (Technical Report No. 29). University of California, Santa Barbara: Project TEECh.

Cosden, M.A., Gerber, M.M., Semmel, D.S., Goldman, S.R., & Semmel, M.I. (1987). Microcomputer use within microeducational environments. Exceptional Children, 53(5), 399-409.

Gerber, M.M. (1986, April). Effects of errorless and repeated practice using microcomputer-administered spelling dictation for learning handicapped students. (Technical Report No. 31). University of California, Santa Barbara: Project TEECh.

Gerber, M.M., English, J.P., & Semmel, M.I. (1985). Microcomputer-administered spelling tests: Effects on learning handicapped and normally achieving students. Journal of Reading, Writing, and Learning Disabilities International, 1(2), 165-176.

Gerber, M.M., Tan S.L., Rothman, H., & Semmel, D.S. (1987). Effects of Self-Rated Competencies and Handicapped Student Behavior on Teachers' Monitoring and Use of Computers. University of California, Santa Barbara: Project TEECh.

Goldman, S.R. & Mertz, D.L. (1986, January). Developing automaticity in retrieving word meaning: Microcomputer practice by learning handicapped students; (Technical Report No. 24). University of California, Santa Barbara: Project TEECh.

Goldman, S.R., Pellegrino, J.W., Mertz, D.L. (November, 1985). Microcomputer delivery of addition drill and practice to math-disabled learners. Interim report 1 (Technical Report #25.0). Santa Barbara: University of California, Santa Barbara, project TEECh.

Pellegrino, J.W. & Goldman, S.R. (1987), "Information Processing and Elementary Mathematics," Journal of Learning Disabilities, 20 (1), 23-32. 57.

Russell, R.L., & Ginsburg, H.P. (1984). Cognitive analysis of children's mathematics difficulties. Cognition and Instruction, 1(2), 217-244.

Schnorr, J.M., & Semmel, D.S. (1986). Evaluating Drill and Practice Programs. Academic Therapy, 22(2), 141-156.

Semmel, D.S., Goldman, S.R., Gerber, M.M., Cosden, M.A., & Semmel, M.I. (1985). Survey of special education and mainstream teachers' access to and use of microcomputers with mildly handicapped students (Technical Report No. 9). University of California, Santa Barbara: Project TEECh.

Semmel, D.S., Schnorr, J., & McComb, A.E. (1986). Teachers' role in the use of microcomputers in the education of mildly handicapped pupils (Technical paper). University of California, Santa Barbara: Project TEECh.

Shavelson, R., Winkler, J., & Feibel, W. (1985). Patterns of microcomputer use in teaching mathematics and science. Journal of Educational Computing Research, 1(4), 395-413.

Svenson, O., & Broquist, S. (1975). Strategies for solving simple addition problems: A comparison of normal and subnormal children. Scandinavian Journal of Psychology, 16, 143-151.

Bibliography

Ashcraft, M.H. (1985). Children's knowledge of simple arithmetic: A developmental model and simulation. Unpublished manuscript, Cleveland State University.

Ashcraft, M.H., Fierman, B.A., & Bartolotta, R. (1984). The production and verification tasks in mental additions: An empirical comparison. Developmental Review, 4, 157-170.

Becker, H.J. (1986, June). Instructional uses of school computers: Reports from the 1985 national survey (Issue No. 1). Baltimore, MD: John Hopkins University.

Becker, H.J. (1984) School uses of microcomputers: Reports from a national survey. Baltimore, MD: John Hopkins University, Center for Social Organization of Schools.

Carpenter, T.P. (1985). Learning to add and subtract: An exercise in problem solving. In E.A. Silver (Ed.), Teaching and learning mathematical problem solving: Multiple research perspectives (pp. 17-40). Hillsdale, NJ: Lawrence Erlbaum Associates.

Christensen, C. A., & Gerber, M.M. (1986, March). Effects of game format in computerized drill and practice on development of automaticity in single digit addition for learning disabled students. (Technical Report No. 29). University of California, Santa Barbara: Project TEECh.

Conner, F.P. (1983). Improving school instruction for learning disabled children: The Teachers College Institute. Exceptional Education Quarterly, 4(1), 23-44.

Cosden, M.A., Gerber, M.M., Semmel, D.S., Goldman, S.R., & Semmel, M.I. (1987). Microcomputer use within microeducational environments. Exceptional Children, 53(5), 399-409.

Finn, J.D., & Resnick, L.B. (1984). Issues in the instruction of mildly mentally retarded children. Educational Researcher, 13(3), 9-11.

Fleishner, J.E., Garnett, K., & Sheperd, M.J. (1980). Basic number fact performance of learning disabled and nonlearning disabled

children. New York: Teachers College, Columbia University, Institute for Study of Learning Disabilities.

Fuson, K.C. (1982). An analysis of the counting on procedure in addition. In T.P. Carpenter, J.M. Moser, & T.A. Romberg (Eds.), Addition and subtraction: A cognitive perspective (pp. 67-81). Hillsdale, NJ: Lawrence Erlbaum Associates.

Fuson, K.C., & Hall, J.W. (1983). The acquisition of early number word meanings: A conceptual analysis and review. In H.P. Ginsburg (Ed.), The development of mathematical thinking (pp. 49-107). New York: Academic Press.

Gelman, R., & Gallistel, C.R. (1978). The child's understanding of number. Cambridge, MA: Harvard University Press.

Gerber, M.M. (1986, April). Effects of errorless and repeated practice using microcomputer-administered spelling dictation for learning handicapped students. (Technical Report No. 31). University of California, Santa Barbara: Project TEECh.

Gerber, M.M., English, J.P., & Semmel, M.I. (1985). Microcomputer-administered spelling tests: Effects on learning handicapped and normally achieving students. Journal of Reading, Writing, and Learning Disabilities International, 1(2), 165-176.

Gerber, M.M., Tan S.L., Rothman, H., & Semmel, D.S. (1987). Effects of Self-Rated Competencies and Handicapped Student Behavior on Teachers' Monitoring and Use of Computers. University of California, Santa Barbara: Project TEECh.

Goldman, S.R. & Mertz, D.L. (1986, January). Developing automaticity in retrieving word meaning: Microcomputer practice by learning handicapped students; (Technical Report No. 24). University of California, Santa Barbara: Project TEECh.

Goldman, S.R., Mertz, D.L., & Pellegrino, J.W. (1986, January). Microcomputer delivery of addition drill and practice to math disabled learners: Interim Report II. (Technical Report No. 26. University of California, Santa Barbara: Project TEECh.

Goldman, S.R., & Pellegrino, J.W. (1986, August). Effective drill and practice on the microcomputer. Santa Barbara: University of California, Santa Barbara, Department of Education.

Goldman, S.R., Pellegrino, J.W., Mertz, D.L. (November, 1985). Microcomputer delivery of addition drill and practice to math-disabled learners. Interim report 1 (Technical Report #25.0). Santa Barbara: University of California, Santa Barbara, project TEECh.

Groen, G.J., & Parkman, J.M. (1972). A chronometric analysis of simple addition. Psychological Review, 79, 329-343.

Hamann, M.S., & Ashcraft, M.H. (1985). Simple and complex mental addition across development. Journal of Experimental Child Psychology, 40, 49-72.

Houlihan, D.M., & Ginsburg, H.G. (1981). The addition methods of first- and second-grade children. Journal for Research in Mathematics Education, 12(2), 95-106.

Pellegrino, J.W. & Goldman, S.R. (1987), "Information Processing and Elementary Mathematics," Journal of Learning Disabilities, 20 (1), 23-32. 57.

Pellegrino, J.W., & Goldman, S.R. (1986, March). Information processing and elementary mathematics (Technical Report NO. 20). Santa Barbara: University of California, Santa Barbara, Project TEECh.

Resnick, L.B. (1982). Syntax and semantics in learning to subtract. In T,P, Carpenter, J.M. Moser, & T.A. Romberg (Eds.), Addition and subtraction: A cognitive perspective (pp. 136-155). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Russell, R.L., & Ginsburg, H.P. (1984). Cognitive analysis of children's mathematics difficulties. Cognition and Instruction, 1(2), 217-244.

Schnorr, J.M., & Semmel, D.S. (1986). Evaluating Drill and Practice Programs. Academic Therapy, 22(2), 141-156.

Semmel, D.S., Goldman, S.R., Gerber, M.M., Cosden, M.A., & Semmel, M.I. (1985). Survey of special education and mainstream teachers' access to and use of microcomputers with mildly handicapped students (Technical Report No. 9). University of California, Santa Barbara: Project TEECh.

Semmel, D.S., Schnorr, J., & McComb, A.E. (1986). Teachers' role in the use of microcomputers in the education of mildly handicapped pupils (Technical paper). University of California, Santa Barbara: Project TEECh.

Shavelson, R., Winkler, J., & Feibel, W. (1985). Patterns of microcomputer use in teaching mathematics and science. Journal of Educational Computing Research, 1(4), 395-413.

Siegler, R.S., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? In C. Sophian (Ed.), Origins of cognitive skills (pp. 229-293). Hillsdale, NJ: Erlbaum.

Siegler, R.S., & Taraban, R. (1986). Conditions of applicability of a strategy choice model. Cognitive Development, 1, 31-51.

Starkey, P., & Gelman, R. (1982). The development of addition and subtraction abilities prior to formal schooling in arithmetic. In T.P. Carpenter, J.M. Moser, & T.A. Romberg (Eds.), Addition and subtraction: A cognitive perspective (pp. 99-116). Hillsdale, NJ: Erlbaum.

Steffe, L.P., Thompson, P.W., & Richards, J. (1982). Children's counting in arithmetical problem solving. In T. P. Carpenter, J.M. Moser, & T.A. Romberg (Eds.), Addition and subtraction: A cognitive perspective (pp. 83-97). Hillsdale, NJ: Erlbaum.

Svenson, O. (1975). Analysis of time required by children for simple additions. Acta Psychologica, 39, 289-302.

Svenson, O., & Broquist, S. (1975). Strategies for solving simple addition problems: A comparison of normal and subnormal children. Scandinavian Journal of Psychology, 16, 143-151.

Svenson, O., & Hedenborg, M.L. (1979). Strategies used by children when solving simple subtractions. Acta Psychologica, 43, 477-489.

Torgesen, J.K., & Young, K.A. (1983). Priorities for the use of microcomputers with learning disabled children. Journal of Learning Disabilities, 16(4), 234-237.

Woods, S.S., Resnick, L.B., & Groen, G.J. (1975). An experimental test of five models for subtraction. Journal of Educational Psychology, 67, 17-21.

MICROCOMPUTER-BASED ASSESSMENT OF THE SPEED OF RETRIEVAL OF MULTIPLICATION FACTS, EXTENDED PRACTICE AND THE DEVELOPMENT OF AUTOMATICITY

Michael M. Gerber

Melvyn I. Semmel

Dorothy S. Semmel

University of California
Santa Barbara

Joan Karp

University of Minnesota
Duluth

Support for this research was provided by a grant from the USDOE/OSERS, to IPA, Associates, Santa Barbara, CA. The opinions expressed do not necessarily reflect the positon or policy of the granting agency and no official endorsement should be inferred.

RUNNING HEAD: MICROCOMPUTER-BASED DEVELOPMENT OF AUTOMATICITY

Learning disabled students' difficulty in higher order learning has been attributed to their failure to automatize simple components of complex tasks in, for example, reading and mathematics (LaBerge & Samuels, 1974; Sternberg & Wagner, 1982; Wolf, Bally, & Morris, 1986). However, the theoretical construct of "automaticity" invoked by these and other investigators derives mostly from laboratory work conducted to study memory (e.g., Ceci, 1984; Shiffrin & Schneider, 1977). Relatively little research of this type has been conducted on applied school tasks (Farnham-Diggory, 1986). Some studies have been conducted that relate developmental shifts in strategy for retrieval of simple arithmetic facts to decreasing latency of response (e.g., see Ashcraft, 1987; Goldman, Mertz, & Pellegrino, 1989; Groen & Parkman, 1972; Siegler, 1987). Others have attempted to establish normative speeds of retrieval in basic arithmetic (Goldman & Mertz, 1985; Sigler & Schrager, 1984). These studies generally focus on mental addition and subtraction, operations for which there are well developed cognitive and chronometric models. Very few studies have addressed speeded retrieval in multiplication or division. In fact, relatively little applied research has focused on how to establish automaticity of simple arithmetic in students with learning difficulties, or how increased retrieval speeds affect performance on hierarchically more difficult tasks.

Microcomputer-Based Practice

The development of microcomputer-based mathematics drill and practice programs promises an efficient method for providing students with extended practice to increase retrieval speed. Initial efforts to investigate effects of computer-based practice on accuracy and speed have yielded mixed findings. Chiang (1986), for example, demonstrated the effectiveness of daily, 4-minute drill trials on speed of written multiplication responses for six, 4th grade learning disabled students. Campbell et al. (1987), however, found no reliable differences on 3rd grade students' division performance resulting from daily, 20-minute microcomputer or paper and pencil drill trials. Drilling 9 to 12 year old learning disabled students in natural school settings, Goldman and Pellegrino (1987) found that extended microcomputer drill in addition facts also yielded mixed results. When high levels of accuracy were maintained, one sample of students showed statistically significant pre-post keystroke gains, but no gains in oral response latency Another group showed no significant gains

on keystroke latencies, but did improve on oral latencies. Goldman and Pellegrino's studies are especially interesting because they call attention to the considerable variability in observed performance attributable to uncontrolled schedules and quality of practice trials in participating schools, as well as individual differences in computer experience, aptitude and prior achievement among participating students. Nevertheless, when comparing their students observed gains to theoretical models relating latency to addition strategy, Goldman and Pellegrino speculated that extended practice, per se, may not be sufficient to promote automaticity in LD students even when accuracy is reliably high.

Microcomputer-based drill and practice programs also provide a means for effectively controlling instructional features of drill (e.g., set composition, set size, response tempo) as well as measuring latency. These instructional features may facilitate or inhibit development of automaticity. Christensen and Gerber (in press), for example, compared a popular, arcade-game type drill program with a "plain vanilla" version of their own design. LD students showed significant speed gains in keyboard, oral, and written responding using both programs. However, LD students, as opposed to normally achieving peers, were consistently disadvantaged by the arcade-game version.

In a recent study, Hasselbring et al. (1988) also investigated effects of extended microcomputer-based drill and practice of basic addition and subtraction on development of automaticity by LD students. However, these researchers tailored their program to explicit instructional criteria, including

1. sorting of practice items by measured latency of keyboard response on a computer-based pretest

2. systematic introduction of new items for practice based on size of smaller addend

3. limitation on size of practice set

4. use of "controlled response time" , or a limit on time permitted for students to respond.

5. interspersal of fast (i.e., "automatic") and slower facts in the same practice set

In the study reported by Hasselbring et al. (1988), LD students practiced for about 10 minutes per day for 49 days. Results showed a 73% increase in the number of rapidly recalled facts. Thus, LD students achieved what these authors regard as automatic retrieval for 45 of the possible 100 (0 to 9 addends) addition facts, a level approximately equivalent to that of non-practicing, nonhandicapped peers.

Despite significant results, it is unclear from this and previously discussed studies how "automaticity" should be characterized operationally. We cannot ascertain whether practicing students' responses become automatic or whether they merely become faster at employing non-automatic strategies. Further, we do not know if student's attainment of rapid responding to single digit math facts improves their performance on hierarchically more difficult problems (ie, multidigit problems with and without added operations). The research reported here was part of a larger research program aimed at addressing these latter questions. For the present study, we designed a practice program that incorporates algorithms that collectively and operationally define automaticity of multiplication fact retrieval. Specifically, the effects of using this program on performance by low achieving intermediate grade students was examined.

Methods

Subjects

Fourth through sixth grade students from the same school in a moderately sized, mid-income California school district participated in this study. Thirty-four students who were relatively low performing in mathematics compared to their classroom peers and who were perceived as needing "extended practice of basic mathematics facts on a computer to increase their speed" were nominated by their teachers. The total number of students was divided about evenly between boys (18) and girls (16). A small number of students were non-native speakers of English (3) or receiving special education resource program services (7). Standardized mathematics test

scores available for inspection validated teacher's judgements (Mean = 45%ile , SD = 24).

Setting and general procedures

All practice trials were held in a normal size classroom used by the participating school as a microcomputer laboratory. The laboratory consisted of sixteen Apple IIGS microcomputers with standard features arranged as a series of open work stations. Students were scheduled to participate in the automaticity practice trials three times each week with each trial consisting of two lessons of twenty problems. Trials typically lasted about ten minutes. All of the students had had a variety of experiences using computers in this laboratory as part of their normal school program and, therefore, had considerable experience in using commercial software. Each student used their own copy of the practice program provided on 3.5" floppy diskettes.

Program features

The program used in this study was specifically designed by the authors to enhance development of automaticity of basic math skills (Semmel, D. S., Semmel M. I., & Gerber, M. M., 1990). Generally, the visible aspects of the program consist of vertically oriented multiplication facts displayed on the screen in large (i.e., about 48 pt.) font. Students press the number key, either on the top keyboard row or on the number pad, that corresponds to the product. Screen presentation is timed for decisions internal to the program, but exposure time is not arbitrarily limited. Students press the RETURN key when they are ready for the next multiplication.

The program itself consists of three major related components: a placement assessment, a series of practice item sets, and a series of measurement and decision algorithms.

Placement assessment. There are two major elements to the computer-based placement assessment. The first element samples basic facts from a 10 x 10 matrix of addition, subtraction, multiplication, or division facts. In the present study, the sampling matrix consisted of all multiplication facts with

multipliers and multiplicands between 0 and 9. The sampling algorithms employed by the program "test" logical likelihood of students' accuracy on facts by sampling items with the same and incrementally larger or smaller multipliers. The output of this sampling process is a "placement" level, or initial practice set.

The second element of the placement pretest has students respond to display of single digit numbers by pressing the key for that number. This produces the "initial keypress latency" (IKL), a measure intended to reflect a response requiring little information processing. The IKL established a recognition, scanning, and motor speed baseline used to "calibrate" other latency measurement algorithms.

Practice item sets. Following completion of the assessment, the program scheduled practice for students at one of ten "levels." Levels were defined as all items from the multiplication matrix that had the same multiplier, and sequenced ordinally. Thus, items like 5 x 3, 5 x 7, 5 x 9, etc. are all pooled at the same level, while items like 4 x 3, 4 x 7, and 4 x 9 are pooled at a different, "lower" level. Each practice set consisted of two, random presentations of each item minus any item that students repeatedly missed on the initial assessment.

Measurement and decision algorithms. Keypress latencies, in milliseconds, were measured by the Apple IIGS internal clock as the time from initial screen display to final keyed response. Latencies for each response and a calculated median and fastest latency of response during each practice trial were stored both as data and as variables for subsequent program decision-making.

The core algorithms operating in the program concern decisions about whether demonstrated speeds of response are likely to be as fast as possible -- that is, automatic. When an affirmative decision is made, students progress to the next higher level of multiplication items. When the decision is negative, another practice trial at the same level is internally scheduled. This decision process occurs within the program after every completed trial. Thus, the actual number of practice trials required by any given student at a given level varies

as a function of that student's unique history of performance as measured by the program.

In essence, the automaticity decision algorithms represent tests of four criteria. First, the program questions time (number of trials) in level. Retrieval time as well as computer measures of time are subject to error due to transient influences on attention. Therefore, students must complete at least four trials.

Second, the program checks to make sure that 80% or better accuracy has been maintained. Students must practice correct responses and only latencies for correct responses can be meaningfully interpret in terms of developing automaticity.

Third, the median latency for correct responses from the current practice trial must be as fast or faster than the median of pooled latencies composed from the lower 50%ile of previous practice trials. Because attention, scanning, and motor fluctuations influence the right (i.e., high) hand portion of latency distributions, a comparison standard is created by incrementing only from latencies on the left (i.e., low) side of the distribution.

Fourth, the standard deviation of latencies in a given trial must be as small or smaller than the standard deviation of pooled latencies composed from the lower 50%ile of previous practice trials. If individuals differ meaningfully, automaticity represents a limit or boundary condition that is difficult to establish on the basis of central tendency measures. Reasonably, such a limit would be represented not only by unchanging mean or median latencies, but by high and unchanging similiarity (i.e., leptokurtosis) among latencies.

Thus, in this study students were considered "automatic" when they simultaneously could demonstrate sufficient practice, sufficient accuracy, and speed gains associated from reduction of variability.

Research design and measures

Individual differences in practice level, number of practice trials required, and practice curves for latencies were all expected and theoretically pertinent in the present study. Therefore, each student served as their own control; initial and early latency measures served as baselines against which speed displayed in later trials were compared. To provide a degree of comparability, however, all students were scheduled to practice a total of 24 lessons even if they practiced at different levels for different lengths of time.

The program measured, stored, and statistically analyzed keystroke latencies and accuracy for each item in each trial for later retrieval. Keystroke latencies were measured as time elapsed between screen display and final keyed response rounded to one-hundredth of a second.

In addition, a series of three, thirty problem, paper and pencil probes for transfer were administered as a pretest and following the sixth, twelfth, eighteenth, and twenty-fourth practice trials. Probes consisted of two x one digit multiplication problems that varied on the number of procedural steps (i.e., carrying, regrouping) necessary to compute products. The first probe set required no procedural steps, the second required one step, and the third required two procedural steps.

To assure that students were, in fact, reliably accurate in multiplication, and to test the validity of the placement assessment component, all students were administered an untimed test of all one hundred combinations of single digit number facts prior to beginning computer practice trials.

Results

Pretesting showed that students participating in this study were reliably accurate (Mean = 94%, SD = 9%) on the hundred basic multiplication facts and, therefore, suitable candidates for automaticity training. Initial keystroke latencies (keyfinding without computation) averaged 1.50 seconds (SD = .26). Practice results for the 34 students varied as expected. However, overall results showed a generalized pattern of low error and decreased keyboard

response latency over trials. No performance differences were attributable to gender or students' special education status. This section will present a summary of these general findings.

Accuracy

Most students were initially placed a the "nines" level of practice. A very few were reassigned to a lower level when accuracy seemed low or too variable. All students tended to display high rates (i.e., over 80%) accuracy as intended by the structure of the program. Early trials showed greater variability than later trials, but variability decreased and accuracy increased over trials.

Speed Gains

Practice curves for all students showed decrease in latency over trials, although level, rate, and pattern of decrease varied across students. Approximately one-third of the students failed to meet the criteria for automaticity after 24 trials, although these students all showed gains in response speed.

A one factor, repeated measures analysis of variance ($F = 17.61$, $df = 23$, $p = .0001$) showed statistically significant gain in median response speed over trials. Scheffe's procedure for post-hoc comparisons showed significant differences between the first trial. the fourth trial, the sixth trial, and all subsequent trials after the eighth practice trial. Median latency (averaged across students) on Trial 1 was 2.33 seconds (SD= .52). Median latencies decreased to 1.67(.50), 1.46 (.41), 1.32 (.40). and 1.26 (.35) on Trials 6, 12, 18, and 24, respectively. Thus, by trial 12, median fact retrieval speed was essentially the same as prepractice keyfinding (IKL) speed without computation. Figure 1 shows the decline in median latency over all 24 trials. While students as a group showed a median decrease in latency of about one second by Trial 18 (.70 seconds by Trial 6), individual gains by Trial 24 varied from no gains for six of the students to half a second for 7 students to over one and a half seconds for 6 students. Figure 2 is a histogram showing the distribution of students who displayed this range of median speed gains.

Transfer

A one factor, repeated measures analysis of variance showed that students produced statistically significantly more correct answers per minute ($F = 26.06$, $df = 4$, $p = .0001$) in later compared to pretest probes of simple multiplication with no procedural steps. Scheffe's procedure for post-hoc comparisons showed that speed on the third and fourth probes (i.e. after trials 18 and 24) were greater than pretest.

A similar analysis was performed for each of the other probe tests. For one procedural step probes, solution speed did not significantly increase as a function of probe occasions ($p=.08$). However, a significant change in speed did occur on the two procedural step probe items ($F = 3.78$, $df = 4$, $p = .007$). Scheffe's procedure for post-hoc comparison showed that speed of producing correct solutions was significantly faster on the last (i.e. after trial 24) compared with the first (i.e., after trial 6) probe. Figure 3. presents a graphical summary of pretest and probe outcomes.

Discussion

In this study we specified an operational definition of automaticity of simple math fact retrieval in the context of a computer program and attempted to estimate its effectiveness for intermediate elementary students who performed poorly in arithmetic in two ways. First, we observed effects of practice on latency of correct multiplication responses in thirty-four individual cases. Second, we attempted to assess what facilitative effect might exist on multiplication problems that required increasing procedural, as opposed to computational, effort as a function of speeded retrieval of simple facts.

Despite the fact that individual patterns of gain differed, especially with respect to inter-trial variability, almost all students showed gains in median latency of 500 milliseconds or more. Median correct response latencies in later trials compared favorably with median latencies for simple key finding (IKL). Thus, students were retrieving and finding appropriate keys to respond in about the same time it took them earlier to simply find keys corresponding to numbers displayed randomly on the screen.

Speed gains on simple facts correlated with gains in speed of accurate solution of more complicated multiplication when that multiplication required no procedural knowledge or effort. The evidence from both one step and two step procedural probes might be interpreted to show a facilitative effect of speeded retrieval of basic facts, but such a conclusion is premature. All probes were repeated a total of five times. Computational aspects of these tasks may have improved as a simple function of repeated exposure and practice rather than as a consequence of developing automaticity of fact retrieval. If this explanation is valid, it would be expected that "no step" problems would improve at a faster rate than "step" problems. The data were consistent with this expectation. For although the rate of correct problem solving was found to be higher between the 6th and 24th trials on the two step transfer problems, the practical significance of this outcome is diminished by the fact that students were working at an extremely slow rate. In general, when employment of added non-computational procedures were necessary, gains in speed of correct responding were small.

References

Ashcraft, M. H. (1987). Children's knowledge of simple arithmetic: A developmental model and simulation. In J. Bisanz, C. Brainerd, & R. Kail (Eds.), Formal methods in developmental psychology (pp. 302-338). New York: Springer-Verlag.

Campbell, D. L. (1987). A comparison of third-grade student performance in division of whole numbers using a microcomputer drill program. Computers in the Schools, 4(2), 79-89.

Chiang, B. (1986). Initial learning and transfer effects of microcomputer drills on LD students' multiplication skills. Learning Disability Quarterly, 9, 118-123.

Christensen, C. A., & Gerber, M. M. (In press). Effects of computerized drill and practice games in teaching basic math facts. Exceptionality, 1.

Farnham-Diggory, S. (1986). Time, now, for a little serious complexity. In S. Ceci (Ed.), Handbook of cognitive, social, and neuropsychological aspects of learning disabilities (Vol. 1)(pp. 123-158). Hillsdale, NJ: Erlbaum.

Goldman, S. R., Mertz, D. L., & Pellegrino, J. W. (1989). Individual differences in extended practice functions and solution strategies for basic addition facts. Journal of Educational Psychology.

Goldman, S. R., & Pellegrino, J. W. (1987). Information processing and educational microcomputer technology: Where do we go from here? Journal of Learning Disabilities, 20(3), 144-154.

Hasselbring, T. S., Goin, L. I., & Bransford, J. D. (1988). Developing math automaticity in learning handicapped children: The role of computerized drill and practice. Focus on Exceptional Children, 20(6), 1- 7.

Pellegrino, J. W., & Goldman, S. R. (1989). Mental chronometry and individual differences in cognitive processes: Common pitfalls and their solutions. Learning and Individual Differences.

Semmel, D.S., Semmel, M. I., & Gerber, M. M. (1990). AutoMath: Basic math facts extended practice program. Santa Barbara CA, Instructional Programming Associates.

Siegler, R. S. (1987). The perils of averaging data over strategies: An example from children's addition. Journal of Experimental Psychology:General, 116, 250-264.

Siegler, R. S., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? In C. Sophian (Ed.), Origins of cognitive skills (pp. 229-293). Hillsdale, NJ: Erlbaum.

Wolf, M., Bally, H., & Morris, R. (1986). Automaticity, retrieval process, and reading: A longitudinal study in average and impaired readers. Child Development, 57, 988-1000.

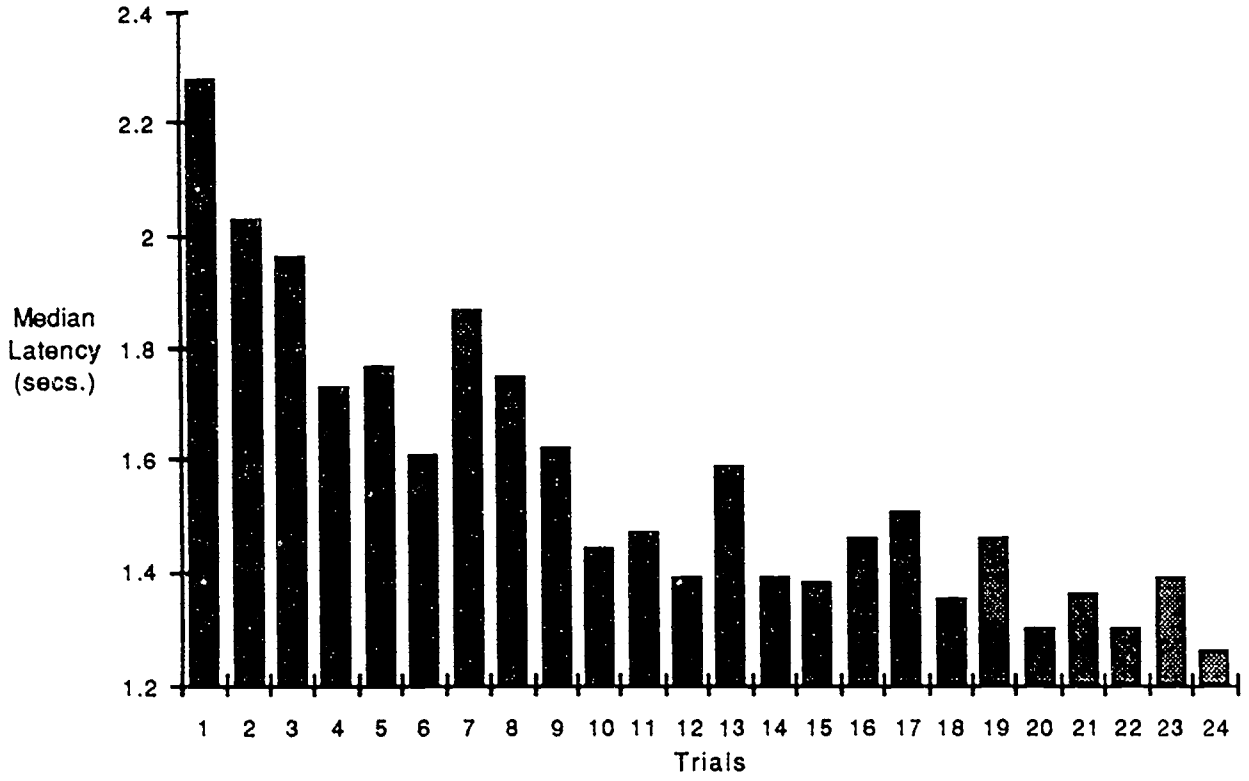


Figure 1. Changes in median latency as a function of practice trial

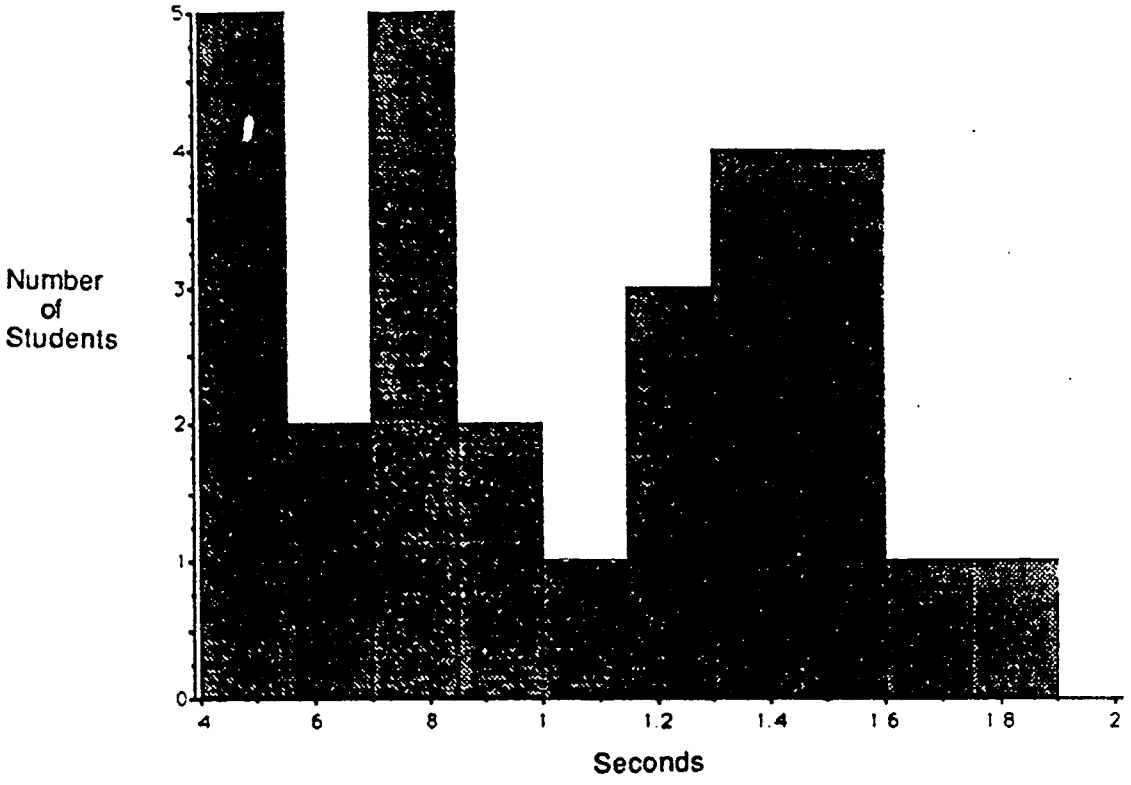


Figure 2. Frequency histogram of gains in response latency over trails

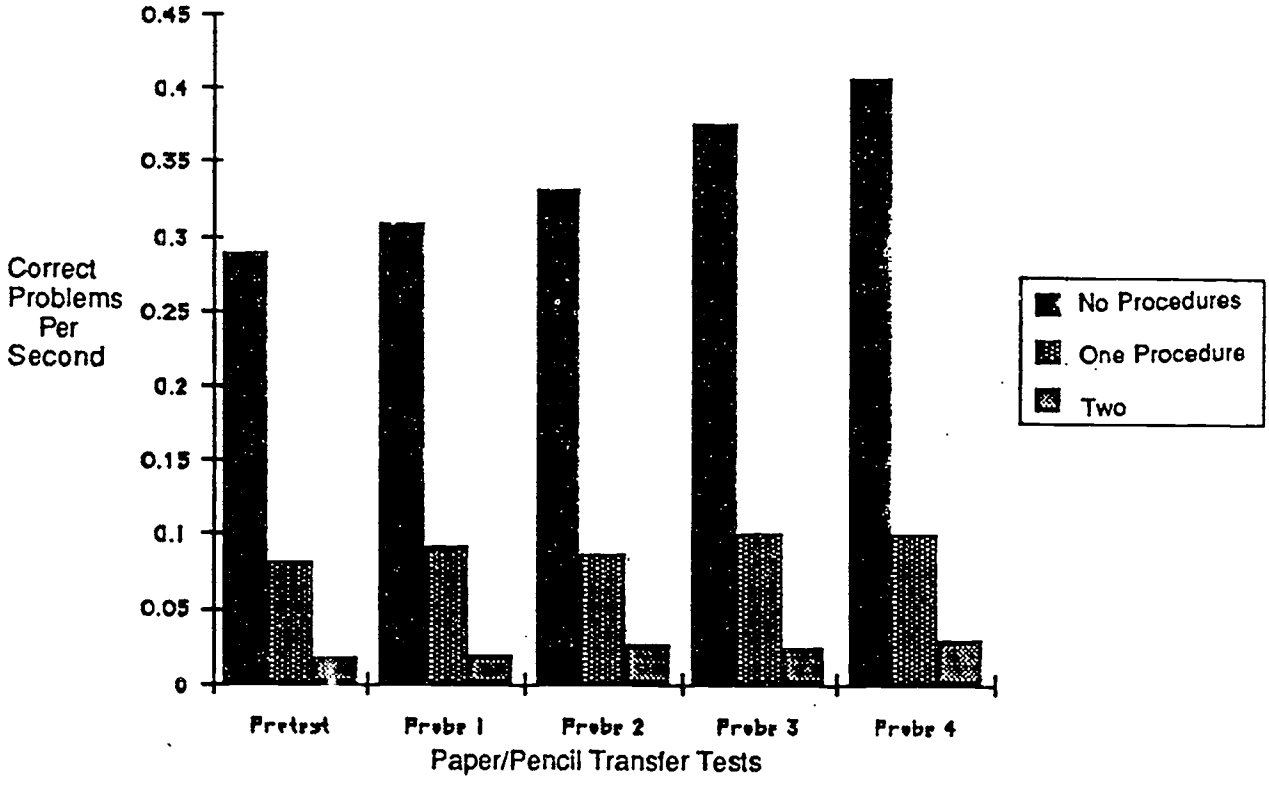


Figure 3. Number of correct problems per minute on paper and pencil multidigit pretest and probes